Universitat Politècnica de Catalunya

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

Grup de Processament de la Imatge

Class Weighted Convolutional Features for Image Retrieval

Albert Jiménez Sanfiz

Advisors: Xavier Giró-i-Nieto and José M. Álvarez López

A thesis submitted in fulfillment of the requirements for the degree of the Master in Telecommunications Engineering

Barcelona, July 2017

Abstract

Image retrieval in realistic scenarios targets large dynamic datasets of unlabeled images. In these cases, training or fine-tuning a model every time new images are added to the database is neither efficient nor scalable. Convolutional Neural Networks trained for image classification over large datasets have been proven effective feature extractors when transferred to the task of image retrieval. The most successful approaches are based in encoding the activations of convolutional layers as they convey the image spatial information.

Our proposal goes beyond and aims at a local-aware encoding of these features depending on the predicted image semantics, with the advantage of using only of the knowledge contained inside the network. In particular, we employ Class Activation Maps (CAMs) to obtain the most discriminative regions of the image from a semantic perspective. Additionally, CAMs are also used to generate object proposals during an unsupervised re-ranking stage after a first fast search.

Our experiments on two public available datasets for instance retrieval, Oxford5k and Paris6k, demonstrate that our system is competitive and even outperforms the current state-of-the-art when using off-the-shelf models trained on the object classes of ImageNet.

Keywords: Image Retrieval, Visual Instance Search, Deep Learning, Convolutional Neural Networks, Transfer Learning.

Resum

La cerca d'imatges en escenaris realistes, es realitza sobre bases de dades dinàmiques on les imatges no estàn etiquetades. En aquests casos, entrenar o tunejar un model cada vegada que noves imatges són afegides a la base de dades, no és ni eficient ni escalable. Les xarxes neuronals convolucionals entrenades amb un gran volum d'imatges per la tasca de classificació han demostrat ser bones extractores de característiques quan es transfereixen a la tasca de cercar imatges similars. Les tècniques proposades amb més èxit, estàn basades en codificar les activacions de capes convolucionals, perquè en elles està continguda la informació espacial de la imatge.

La nostra proposta va més enllà, i té la intenció de codificar aquestes activacions depenent del contingut semàntic (classes) predites per la xarxa. Tot això, utilitzant només el coneixement inclòs a la xarxa. En particular, fem servir *Class Activation Maps* (*CAMs*) per obtenir les regions més discriminatives de la imatge segons una perspectiva semàntica. Addicionalment, les *CAMs* són utilitzades per generar propostes d'objectes durant una etapa de *re-ranking* no supervisat que té lloc després d'una primera cerca ràpida.

Els nostres experiments realitzats en dos bases de dades públicament disponibles, *Oxford5k* i *Paris6k*, demostren que el nostre sistema és competitiu i que inclús supera l'estat de l'art quan s'utilitzen models pre-entrenats amb *ImageNet*.

Paraules clau: Cerca d'Imatges, Retrobament d'Imatges Similars, Aprenentatge Profund, Xarxes Neuronals Convolucionals, Transferència de Coneixement.

Resumen

La búsqueda de imágenes en escenarios realistas, se realiza sobre bases de datos dinámicas, dónde las imágenes no están etiquetadas. En estos casos, entrenar o tunear un modelo cada vez que se añaden nuevas imágenes, no es ni eficiente ni escalable. Las redes neuronales convolucionales entrenadas con un gran volumen de imágenes para la tarea de clasificación han demostrado ser buenas extractoras de características cuando se transfieren a la tarea de búsqueda de imágenes similares. Las técnicas más exitosas están basadas en codificar las activaciones de capas convolucionales, porqué en ellas está contenida la información espacial de la imagen.

Nuestra propuesta va más allá, y tiene la intención de codificar estas activaciones dependiendo del contenido semántico (clases) predecidas por la red. Todo esto utilizando sólo el conocimiento incluido dentro de la red. En particular, hacemos uso de *Class Activation Maps* (*CAMs*) para obtener las regiones más discriminativas de la imagen según una perspectiva semántica. Adicionalmente, las *CAMs* son utilizadas para generar propuestas de objetos durante una etapa de *re-ranking* no supervisado que tiene lugar después de una primera búsqueda rápida.

Nuestros experimentos realizados en dos bases de datos públicamente disponibles, *Oxford5k* y *Paris6k*, demuestran que el sistema es competitivo y que incluso supera el estado del arte cuando se utilizan modelos pre-entrenados con *ImageNet*.

Palabras Clave: Búsqueda de Imágenes, Recuperación de Imágenes Similares, Aprendizaje Profundo, Redes Neuronales Convolucionales, Transferencia de Conocimiento.

"I, a vegades, contra tot pronòstic, una gran bestiesa capgira allò que crèiem lògic, tot fent evident, que per un moment, ens en sortim."

Acknowledgements

First of all, I want to express my sincere gratitude to my advisors, Xavi and José. Xavi thanks for introducing me to the X-thesis research group and the CV reading group. I believe that you are doing a great job in terms of opening the research world to students and establishing a really nice group to work in. José thanks for receiving me when I just arrived to Australia with only two suitcases and no place to go. Thanks for making me think, for your advices, for the enriching discussions and for the much deserved post-work beers. Thank you both for giving me this opportunity and for pushing me. I have enjoyed every moment here (except the cold nights, cold mornings and paper deadlines ;)).

I want to give thanks to Albert Gil and Josep Pujal from our technical support team at the Image Processing Group at UPC for solving the problems I had before coming here. Besides, I want to thank Eva Mohedano for all the tips and advices and Miaomiao for helpful discussions.

I would like to give special thanks to Cristina, who let me stay in her house when I just arrived. Thanks for softening the great changes that were coming and showing me around. I would also like to thank my housemate Bretto who made me feel extremely welcomed. Thank you for making my stay here much more entertaining, for your crazy ideas, your advices and our philosophical talks at The Front.

Thanks, Drop1s (or should I say DropAI) for having kept us together after 6 hard years of university. Adrià, Víctor, Fullana, Chema and Ferran José Torra (by order of messages sent in the chat group). From the first submarine and class D amplifier to building antennas inside tuppers and listening to birds singing in class. Thanks for the great memorable moments that made this university stage much more enjoyable.

Thanks to all my friends. For the skype calls, the laughs and support in the Canberra's coldest days.

Y por último, pero no por eso menos importante, quiero dar las gracias a mi familia. Gracias por estar siempre cuando os necesito, por el apoyo, los consejos y por aguantarme hasta a 17000km de distancia. Os he echado un poquito de menos ;).

Muchas gracias a todos.

Contents

List of Figures

List of Tables

List of Abbreviations

BOW	**B**ag **o**f **W**ords
CAM	**C**lass **A**ctivation **M**ap
CBIR	**C**ontent **B**ased **I**mage **R**etrieval
CNN	**C**onvolutional **N**eural **N**etwork
GAP	**G**lobal **A**verage **P**ooling
GPU	**G**raphics **P**rocessing **U**nit
OfA	**Of**fline **A**ggregation
OnA	**On**line **A**ggregation
QE	**Q**uery **E**xpansion
ReLU	**Re**ctified **L**inear **U**nit
R-MAC	**R**egional - **M**aximum **A**ctivation of **C**onvolutions
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
TF-IDF	**T**erm **F**requency - **I**nverse **D**ocument **F**requency
VLAD	**V**ector of **L**ocally **A**ggregated **D**escriptors

Chapter 1

Introduction

Image retrieval is the process of searching and fetching images from a dataset. The user of these systems typically provides a query such as a keyword, an image or a caption, and the system is expected to return a set of similar images. Some examples for similarity criteria are based on meta tags, color distribution, textures or shape attributes. Content-based Image Retrieval (CBIR) is related to the latter ones, applying computer vision to retrieve images based on their visual content. Figure 1.1 shows a commercial system of *Search by Image* [1]. This system probably combines a content-based image retrieval criterion, as all images follow a similar color outline, with other criteria such as the predicted keywords or visual diversity among results.

Figure 1.1: Example of Google's *Search by Image* [1]

One of the main challenges in CBIR is finding image representations so that related images have higher similarity score than dissimilar ones. One could think, that a good and simple approach for obtaining similar images may be comparing them by their raw pixel values. However, this approach is not robust to changes in scale, translation or illumination. Besides, it is not efficient, as images have millions of pixels. For these reasons, images need to be characterized with a representation which is invariant to certain transformations. Another important and desirable property for these representations is to be compact, that is, having a small memory footprint. When compactness increases, the storage requirements decrease and, in addition, the search speed increases.

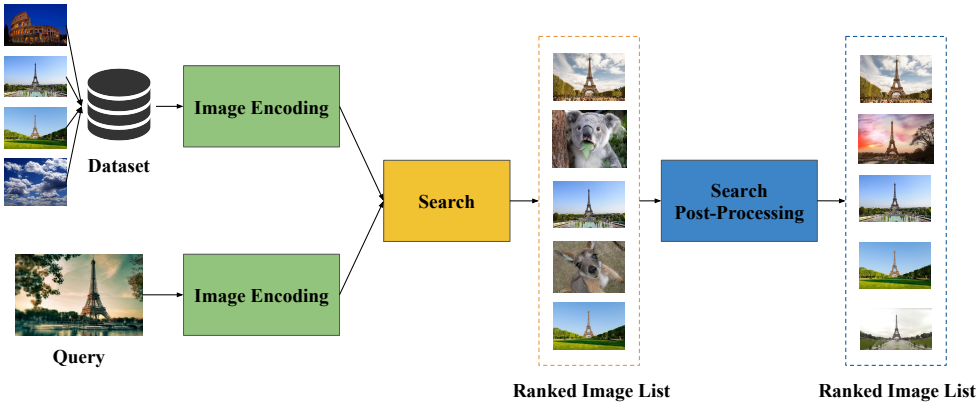


Figure 1.2: Image Retrieval Pipeline.

In general, image retrieval systems first encode and index a dataset of images where the search will be performed. Then, when the user poses a query, a similarity search comparing the query representation with the ones stored in the database is carried out. After ordering the scores computed, a ranked list of images is provided. Finally, a search post-processing is usually applied to refine the initial search. An example of image retrieval pipeline is depicted in Figure. 1.2.

In the first retrieval systems, features were handcrafted to satisfy these invariance properties. More recently, there has been a significant performance improvement by using deep learning models as feature extractors (learned features).

Deep learning [2] is part of a broader family of machine learning methods based on learning representations from data. The most popular architectures employed for dealing with images are called Convolutional Neural Networks (CNNs), as they are based in a set of layers of learnable filters that perform the convolution operation. Since the AlexNet [3] model won the ImageNet object recognition challenge in 2012, the emergence of CNNs revolutionized how the community is tackling computer vision related tasks. When provided with large scale datasets, CNNs have been able to set new state-of-the-art results, outperforming traditional hand-crafted methods. In addition to that, their ability to transfer the features learned from other datasets or tasks has been determining to boost their popularity [4].

The property of transferring knowledge between datasets is particularly important for the image retrieval problem, where the classic study case targets a large and growing dataset of unlabeled images. These images must be efficiently indexed by compact descriptors, which is a costly process that requires exploring the whole database. In this situation, considering an approach where a CNN is re-trained every time new images are added is not efficient and does not scale well. Therefore, many of the works in the literature focus on using pre-trained CNNs as feature extractors or focus on enhancing them by performing a fine-tuning using a custom dataset. For instance, [5] and [6] proposed the use of the activations of the fully-connected layers. Other later works demonstrated that the activations of convolutional layers provide better performance, as they convey the spatial information of the images making them more useful for the task of object retrieval [7]. Following this observation, many works based their approach on combining these convolutional features with regions of interest inside the image [8, 7, 9, 10]. More recently, works have focused on applying supervised learning to fine-tune CNNs using similarity oriented loss functions such as ranking [11] or pairwise similarity [12]. Adapting the CNN boosts the performance of the representations obtained. However, it has the main drawback of having to

spend large efforts on collecting, annotating and cleaning a large dataset, which sometimes is not feasible.

In this thesis, we aim at building better image representations by making use of the image semantics. That includes extracting information from two different layers of a pre-trained network: from a convolutional one and the last fully connected one. Besides, we are not training or fine-tuning a model specifically for the retrieval task or for a particular dataset, we are just taking profit of the built-in network knowledge. We base our argument in the fact that features learned from a large scale dataset [13] can be transferable to any other datasets [4]. The proposed solution can be directly applied to a broad domain of datasets with natural images, with no need of adaptation. The key idea of our approach is obtaining improved image representations by explicitly encoding the spatial information about the objects that appear in the image. There are works like [14] that use both semantic attributes and local features to compute inverted indexes for fast retrieval, or [15] that propose to use an embedding of weak semantic attributes. However, most of the existing retrieval techniques omit computing regions of the image associated with the objects that appear inside, mainly because it relies in other expensive approaches like training an object detector. Our proposal makes use of Class Activation Maps (CAMs) [16], allowing us to leverage the semantic information contained in a pre-trained CNN in a simple fashion.

The main contributions of this work are: First, we propose to encode images based on their semantic information employing CAMs to spatially weight convolutional features. Second, we take advantage of the object mappings given by CAMs to compute fast regions of interest for a posterior re-ranking stage. Finally, we set a new state-off-the art in Oxford5k and Paris6k datasets using off-the-shelf features.

The thesis is organized as follows. First, in Chapter 2, we provide an explanation of the common retrieval pipeline and a review of the state of the art. Then, in Chapter 3, we provide an introduction to Convolutional Neural Networks and its typical layers. In Chapter 4, we review the CAM technique. In Chapter 5, we explain our proposal and introduce our image encoding pipeline. Experiments, ablation studies of our technique and how it compares with the state-of-the-art are described in Chapter 6. In Chapter 7, we summarize the thesis and draw our final conclusions. Finally, appendixes provide additional qualitative results and the accepted version of this work to be presented at BMVC2017 in London.

The source code used for this project can be found in Github:

http://imatge-upc.github.io/retrieval-2017-cam/

Chapter 2

State of the art

The process of image retrieval begins with the exploration of a dataset and the encoding of its images into compact representations. These representations are later compared by means of a distance metric and ranked following the obtained score in order of similarity. After the first search, multiple techniques has been proposed to improve the retrieval performance. This section includes an explanation of the general image retrieval pipeline (Figure 1.2) and a review of the related work in image encoding and search post-processing.

2.1 Image Encoding

2.1.1 Hand-crafted Features

The most successful retrieval approaches before the popularization of Deep Learning were based on locally invariant features [17], often encoded using a Bag of Words model [18] and improved using large visual codebooks [19]. Explained in a nutshell, interest points are detected in the image and local invariant descriptors are extracted. Each descriptor is assigned to its closest visual word in a *visual vocabulary*: a codebook obtained offline by clustering a large set of descriptors with k-means. This results in a typically high dimensional sparse histogram representation. Then, an inverted list structure is employed for efficient indexing and a Term Frequency - Inverse Document Frequency (TF-IDF) scoring is used to discount the influence of visual-words which occur in many images. In Figure 2.1 an example of matching between these descriptors can be seen. Later, other alternatives based on VLAD [20] and Fisher Vectors [21] encoding were proposed as well.

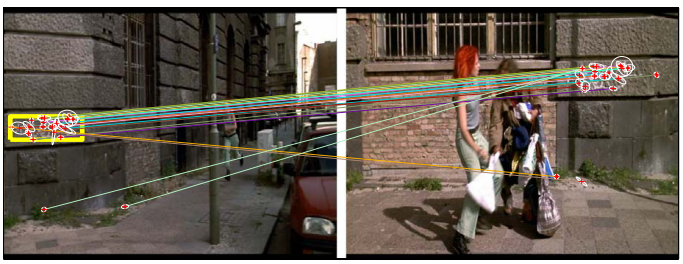


Figure 2.1: Example of SIFT based retrieval [18].

2.1.2 Learned Features

Following the success of CNNs for the task of image classification, recent image retrieval works have replaced the classical hand-crafted features for representations extracted from pre-trained CNNs. If the reader is not familiar with CNNs, in Chapter 3 we provide a review on their fundamentals with some visual examples.

To illustrate better the related work in the matter, we introduce firstly the CNN architecture mostly used in this domain: VGG-16 [22]. This network features a homogeneous architecture based on 3x3 convolutions and 2x2 max-poolings on the convolutional layers, followed by three fully-connected layers and a softmax classifier, as shown in Figure 2.2. A visual explanation of some of the techniques explained below can be found in Figure 2.3.

A first approach to using CNNs for image retrieval was encoding the images using features extracted from the fully connected layers. A high level dense descriptor of the image visual content was obtained, which was referred as Neural Code [5]. It was shown that by means of applying PCA, these codes could be shortened and still performing better than the previous hand-crafted features state-of-the-art. An extension to local analysis was presented in [23], where these features were extracted over a fixed set of regions at different scales defined over the image.

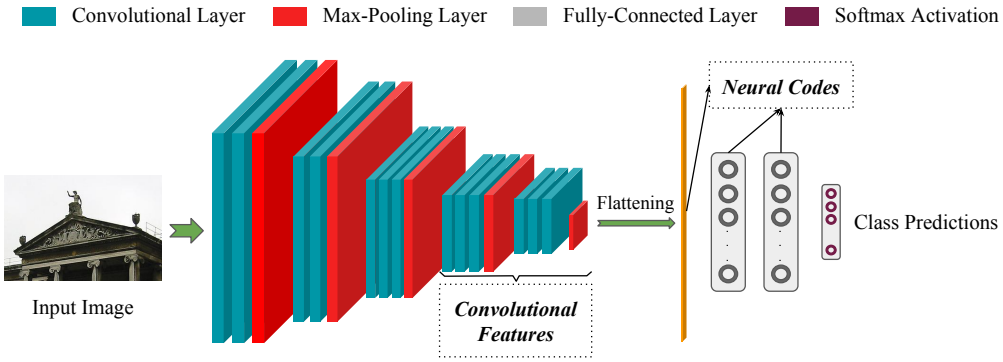


Figure 2.2: VGG-16 Architecture. The first retrieval approaches were based on encoding the images using the activations of the fully connected layers activations (Neural Codes), while the latter ones focused on using the Convolutional Features.

Posterior works observed that features from convolutional layers convey the spatial information of the images, making them more useful for the task of retrieval. In addition to that, it allowed to input variable size images to the network, which also brought an improvement of performance. Based on this observation, different authors have based their approaches on combining convolutional features with different estimation of the areas of interest within the image.

In [7] a global descriptor is built by sum-pooling convolutional features (SPoC descriptor) and introducing a gaussian centering prior, assuming that the relevant content is in the center of the image (introducing a dataset bias). Razavian's technique in [8] considers a multiresolution search, extracting different size sub-patches at random locations. R-MAC [9] used a fixed-rigid grid of different size regions and encode a vector per region by performing max-pooling in every feature map. Then it aggregates each region vector to form a global image representation. In the last place, BoW [24] constructs a Bag of Words model on top of convolutional features using a fixed rigid grid of regions too. These works shows how focusing in local regions of the convolutional features can improve performance, but the computation of these regions is based

on heuristics and randomness, not on the image content.

In this work, we aim at extracting features with focus on local areas that depend on the contents of the image, as other authors have explored in the past. A first option is training a region proposal network for each query object [11, 25], but this solution does not scale well as it is a computational intensive process that must be run at query time, both for the training, and for the analysis of a large scale dataset at search time. A second solution is using an additional model to predict the regions of interest for each image. For example, the work in [26] uses the saliency maps generated by an eye gaze predictor to weight the convolutional features. However, this option requires an additional computation of the saliency maps which duplicates the computational effort of indexing the database. A third approach is proposed by the CroW model [10], which estimates a spatial weighting as a combination of the convolutional feature maps across all channels of the layer. Its authors claim they boost features at locations with salient visual content while down weights in non-salient locations.

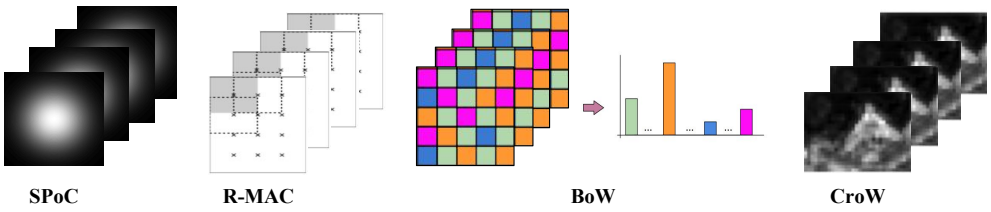


Figure 2.3: A visual comparison of the different techniques proposed. SPoC [7] uses a Gaussian center prior, R-MAC [9] a fixed grid of regions, BoW [24] constructs a visual vocabulary using a grid of regions and CroW [10] boost features at locations with salient content.

This weighting scheme can be efficiently computed in a single forward pass. However, it does not explicitly leverage the semantic information contained in the model. To this end, we adopt the Class Activation Maps (CAMs) proposed in [16] as a method to exploit the predicted classes and obtain semantic-aware spatial weights for convolutional features. Chapter 5 describes the details of this solution and how we have adopted it to the task of image retrieval.

2.2 Search

When the user inputs the system a query image, he expects to receive a set of similar images. The search process compares the input query representation with all the other image descriptors in the dataset. These descriptors are compared and a score is computed using a distance metric. While the first metric used was the Euclidean distance [19], most of the recent works use the cosine similarity. Given two image descriptors X and Y, cosine similarity is computed as follows:

$$Cosine\ similarity\ (X,Y) = \frac{X \cdot Y}{||X||_2||Y||_2} = \frac{\sum_i^n X_i Y_i}{\sqrt{\sum_i^n X_i^2}\sqrt{\sum_i^n Y_i^2}} \tag{2.1}$$

If the descriptors are normalized ($||X||_2 =$, $||Y||_2 = 1$), cosine similarity can be computed in a fast and efficient way by GPUs using the dot product. Furthermore, it is equivalent to compute the Euclidean distance, as shown below:

$$Cosine\ similarity\ (X,Y) = \frac{X \cdot Y}{||X||_2||Y||_2} = X \cdot Y \tag{2.2}$$

$$\begin{aligned} Euclidean\ distance\ (X,Y) &= ||X-Y||^2 = ||X||^2 + ||Y||^2 - 2X^TY \\ &= 2(1 - X^TY) = 2(1 - \ Cosine\ similarity\ (X,Y)) \end{aligned} \tag{2.3}$$

Once the all the scores are computed, they are sorted and the top highest ranked images are returned as the most similar ones.

2.3 Search Post-Processing

Several techniques have been proposed to enhance the performance of retrieval after a first fast search. We can divide them in two different approaches: Query Expansion and Spatial Verification or Re-Ranking.

2.3.1 Query Expansion

Query expansion (QE) is the process of reformulating a query to improve retrieval performance in information retrieval operations. In text retrieval, it could consist of finding synonyms of words, finding all the morphological forms of words, fixing spelling errors or using high ranked documents from the original search to generate a new query that can be used to obtain a new more accurate search. [27] introduces QE to the visual domain, proposes strategies to spatially verify the top-ranked images against query features, in a way that inliers are back-projected by the estimated affine transformation into the query region, and finally, issuing a new query. The differences in the proposed strategies are either in the number of repeated applications of the process, or in the method of feature selection. The quickest and most popular [28, 9, 10, 24] strategy is query averaging, where the new query is obtained by averaging the top-ranked descriptors. It is vital for query expansion that we do not expand using false positives, for this reason, its application is usually preceded by a spatial re-ranking step.

2.3.2 Spatial Re-Ranking

As proposed in in [19], a first fast ranking based on the image features can be improved with a local analysis over the top retrieved images. This re-ranking is based on a more detailed matching between the query object and the location of this object in each top-ranked images. Ideally we would like to verify that the target and query image regions were generated by the same object/scene region. There are multiple ways to obtain object locations For instance, [19, 27] propose to use affine transformations, R-MAC [9] applies a fast spatial search with approximate max-pooling localization. BoW [24] applies re-ranking using a sliding window approach with

variable bounding boxes. In Figure 2.4 we show a visual example of some of the re-ranking techniques proposed.

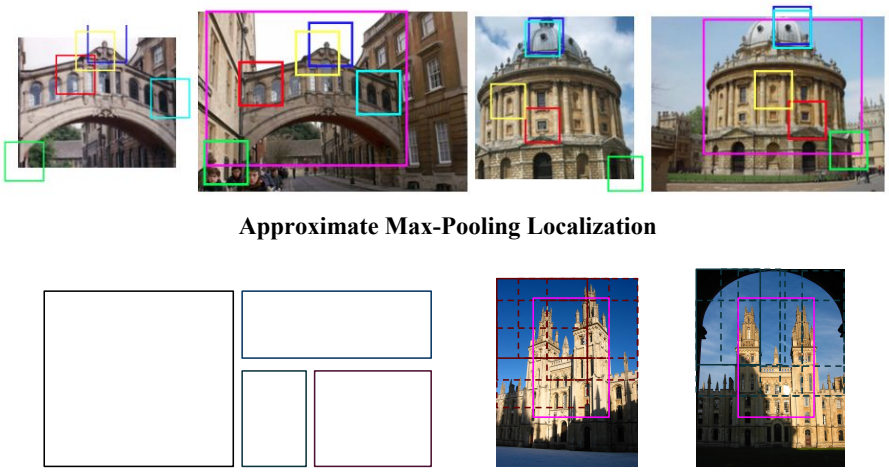


Figure 2.4: Example of two re-ranking techniques. Firstly, the one used by R-MAC [9], each color region corresponds to the maximum response of a feature channel. Secondly, the sliding multiple size sliding windows approach used by BoW [24]. The bounding box in magenta corresponds to the region of interest detected in the image.

Chapter 3

Convolutional Neural Networks

Convolutional networks [29], also known as convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology. Examples of use include time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and image data, which can be thought of as a 2D grid of pixels. CNNs take advantage of the fact that the input consists of a 2D matrix and they constrain the architecture in a different manner than a regular neural network, reducing the number of parameters and decreasing the computational complexity. Convolutional layers have neurons arranged in 3 dimensions: width, height, depth as can be seen in Figure 3.1.

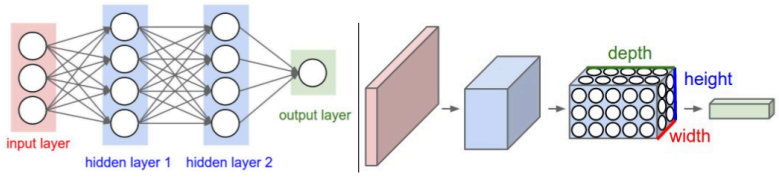


Figure 3.1: Left: A regular 3-layer neural network. Right: A CNN where its convolutional layers arrange its neurons in three dimensions (width, height, depth) [30].

A CNN is composed by a sequence of stacked layers, where every layer transforms one volume (tensor) of activations to another through a differentiable function. Three main types of layers are mostly used:

Convolutional Layers. These layers are composed by a set of learnable filters. Every filter perform the convolution operation with the input image and the result is an activation map per filter as can be seen in Figure 3.2. After this process, we obtain a set of activation maps, also called feature maps, that highlight certain regions of the image (e.g. In Figure 3.4 after the first convolutional layer we have 6 feature maps of dimensions 28×28). Then these feature maps are used as an input for the next layer, being able to capture more complex patterns as the depth increases.

Pooling Layers. These layers perform a downsampling along the spatial dimensions of the input. This operation is done by sliding a window through the input and performing an operation such as taking the maximum value (max pooling) or the average value (average pooling). By pooling, we reduce the spatial resolution, thus decreasing number of parameters to be learn and gaining some translation invariance. Figure 3.3 shows an example of pooling.

Fully-Connected Layers. The high level reasoning in the network is done via these layers. Neurons in these layers have connections to all the previous layer activations, therefore, they have a larger number of parameters. The recent trends in CNN architectures include avoiding their use to decrease the memory requirements. In networks trained for classification, it is common that the last layer is a fully-connected one, followed by a softmax classifier.

In addition to that, activation functions are often plugged into the outputs of the layers such as the Rectifier Linear Unit (ReLU) to introduce non-linearities, or the Softmax to compute probabilities.

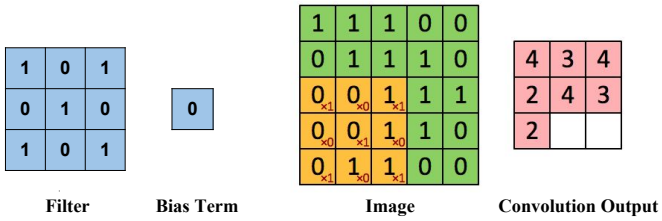


Figure 3.2: Convolution example. The input image is convolved by the filter. The last grid show the activations after convolving for each spatial position.

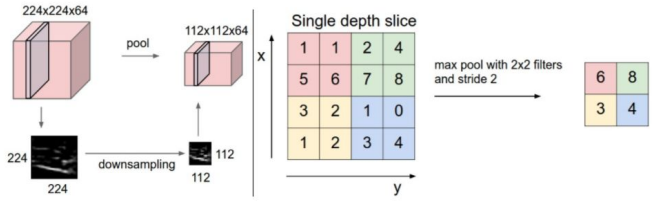


Figure 3.3: Left: In this example, the input volume of size $224 \times 224 \times 64$ is pooled with filter size 2, stride 2 into output volume of size $112 \times 112 \times 64$. Notice that the volume depth is preserved. Right: The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2 [30].

The first successful applications of convolutional networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture [29], shown in Figure 3.4, that was used to read zip codes, digits, etc.

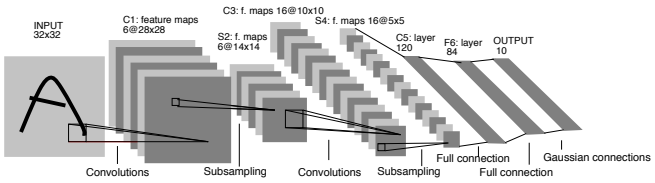


Figure 3.4: LeNet Architecture [29]. Composed by two sets of convolutional, activation, and pooling layers, followed by a fully-connected layer, activation, another fully-connected, and finally a softmax classifier.

Chapter 4

Class Activation Maps

Class Activation Maps (CAMs) [16] were proposed as a method to estimate the pixels of the image that were most attended by the CNN when predicting each semantic class. The computation of CAMs is straightforward in most state-of-the-art CNN architectures for image classification by replacing the last fully-connected layers for a Global Average Pooling (GAP) layer and a linear layer (classifier). In the original work, another convolutional layer before the GAP (*CAM layer*) was added to improve the performance lost after the removal of the fully-connected layers. In the case of having a network architecture like the recent DenseNet [31] or ResNet [32] where the layer before the classifier is a GAP layer, CAMs can be directly extracted without any kind of network modification. The details of our specific architecture can be seen in Section 6.1 and in Figure 4.1.

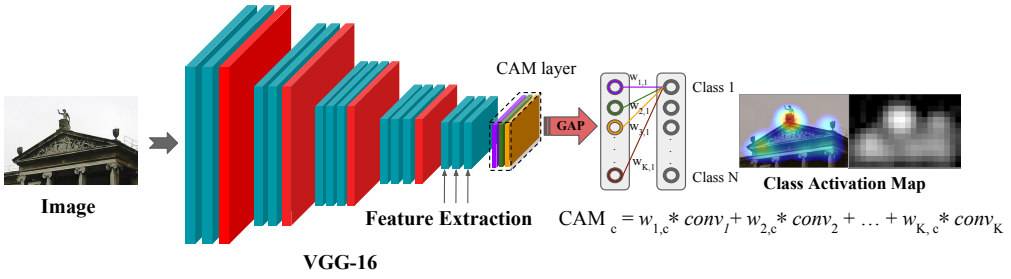


Figure 4.1: Close view of our network architecture and the Class Activation Maps computation. The network is a modified VGG-16 where the last fully-connected layers have been replaced by a convolutional layer (CAM layer), a GAP layer and finally a a linear layer with softmax normalization. We show the normalized CAM generated for that image as well as a heatmap superposed with the original.

Given an output semantic class *c*, its CAM is computed as a linear combination of the feature maps in the last convolutional layer, weighted by the class weights, learned by the linear classifier. More precisely, the computation of the Class Activation Map for the $c-th$ class CAM_c is as follows:

$$CAM_c = \sum_{k=1}^{K} conv_k \cdot w_{k,c} \tag{4.1}$$

where $conv_k$ is the kth feature map of the last convolutional layer before the GAP layer, and $w_{k,c}$ is the weight associated with that feature map k and class c. Notice that, as we are applying a global average pooling before the classifier, the CAM architecture is valid for images of any size.

From CAMs is possible to extract bounding boxes estimating the localization of objects as shown in the Figure 4.2. The procedure consists of setting a threshold based on the normalized intensity of the CAM heatmap values and then setting to zero all the values below that threshold. Then define as a region of interest the bounding box that covers the largest connected element. In section 6.4 we explain how we have adapted this algorithm for our particular use.

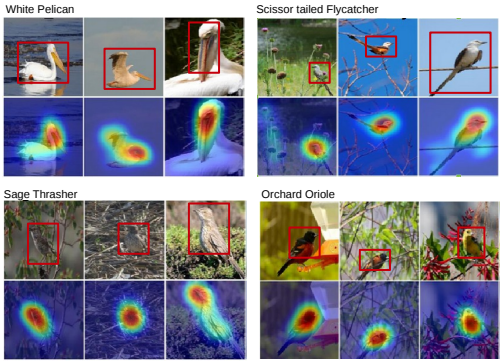


Figure 4.2: CAMs and the inferred bounding boxes (in red) for selected images from four bird categories in CUB200 [16].

Chapter 5

Methodology

The main goal of this thesis is encoding images into compact representations, taking into account the semantics of the scene by using only the knowledge contained in a CNN. To achieve that, we use convolutional features weighted by a soft attention model over the semantic classes detected in the image. Our main contribution is exploiting the transferability of the information encoded in a CNN, not only in its features, but also in its ability to focus the attention on the most representative regions of the image. For this goal, we explore the potential of Class Activation Maps (CAMs) [16] to generate semantic-aware weights for convolutional features extracted from the deeper layers of a network.

In Chapter 4 we provided a review of the CAMs framework. Figure 5.1 presents the whole pipeline of our proposal, which is described in detail in this Chapter. In addition to that, Section 6.2 includes an ablation study of the impact of each block contained in the proposed pipeline.

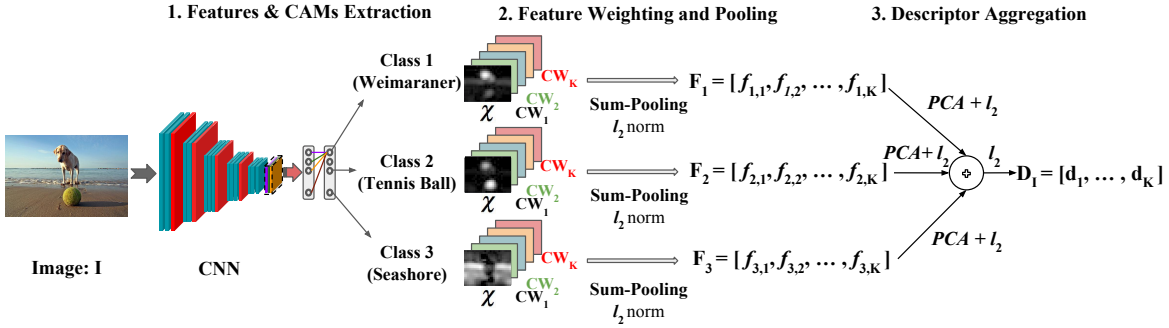


Figure 5.1: Our image encoding pipeline. After forwarding the image, its feature maps are weighted spatially by each CAM, sum-pooled and weighted by channel. Then, PCA and l_2 normalization are applied to each class vector F_c. Finally, they are aggregated and l_2 normalized again to build a compact representation D.

The image encoding pipeline depicted in Figure 5.1 is divided in three stages:

1. Features and CAMs Extraction. Each image is feed-forwarded through the CNN to compute, in a single pass, the convolutional features of a selected layer and the CAMs. The selected convolutional layer has K feature maps (χ) of width W and height H. Every CAM highlights the class-specific discriminative regions attended by the network to make its predictions. CAMs are normalized to fall in the range $[0, 1]$ and, if their dimensions do not match the ones of

the selected convolutional feature maps, they must be finally resized. The details regarding how CAMs are computed can be found in Chapter 4.

2. Feature Weighting and Pooling. Once the convolutional features and the CAMs have been extracted, the next step is weighting the features and pooling them to obtain a compact representation. For a given class c, we weight its features spatially, multiplying element-wise by the corresponding normalized CAM. Afterwards, each convolutional feature map is reduced to a single value by sum-pooling, in such a way that the dimensions of the feature vector corresponds to the amount of convolutional filters in the selected layer. We choose sum-pooling instead of max-pooling because we want to cover the extension of the objects rather than the most discriminative part. Furthermore, sum-pooling aggregation benefits more from the later application of PCA and whitening, as we observed experimentally and equally noted in [7, 10]. Finally, we include the channel weighting proposed in CroW [10] to reduce channel redundancies and augment the contribution of rare features. More precisely, we compute the proportion of non zero responses for each channel with respect to the feature map area Q_k as

$$Q_k = \frac{\sum_{i,j} 1\ [\chi_{i,j}^{(k)} > 0]}{WH}. \tag{5.1}$$

Then, we assign a weight for each channel CW_k,

$$CW_k = \log(\frac{\sum_{n=1}^{K}(Q_n)}{Q_k}) \tag{5.2}$$

that is computed as the logarithm of the inverse channel sparsity. Finally, we obtain a fixed length class vector, $F^c = [f_1^c, f_2^c, ..., f_K^c]$, where K is the depth of convolutional layer. Each of its components is computed as follows:

$$f_k^{(c)} = CW_k \sum_{i=1}^{W}\sum_{j=1}^{H} \chi_{i,j}^{(k)}\ CAM_{i,j}^{(c)} \tag{5.3}$$

3. Descriptor Aggregation. The final step is building a descriptor D_I for each image I by aggregating N_C class vectors. We perform l_2 normalization, PCA-whitening [33] and l_2 normalization once more as in [10, 9]. Then we combine the number of class vectors into a single one by summing them and l_2 normalizing again in the end.

All the classes that we have available to aggregate are given by a pre-trained CNN. As we are transferring the learning into other datasets, we have to define a policy to select which classes are the most relevant. We define two basic approaches depending on the moment when we build the descriptors for the dataset:

Online Aggregation (OnA). The top N_C predicted classes of the query image are obtained at search time (online) and the same set of classes is used to aggregate the features of each image in the dataset. This strategy, while generating descriptors that adapt to the query, presents two important drawbacks which do not make it scalable. First, it requires extracting and storing the CAMs for all classes for every image from the target dataset, with the corresponding requirements in terms of computation and storage. Secondly, the aggregation of weighted feature maps must also be computed at query time, which slows down the retrieval process.

Offline Aggregation (OfA). The considered top N_C semantic classes can also be predicted individually for each image in the dataset at indexing time. This task is performed offline and no

intermediate information needs to be stored, just the final descriptor, which makes the system more scalable than the online approach.

A visual explanation of both strategies can be found in the following figures:

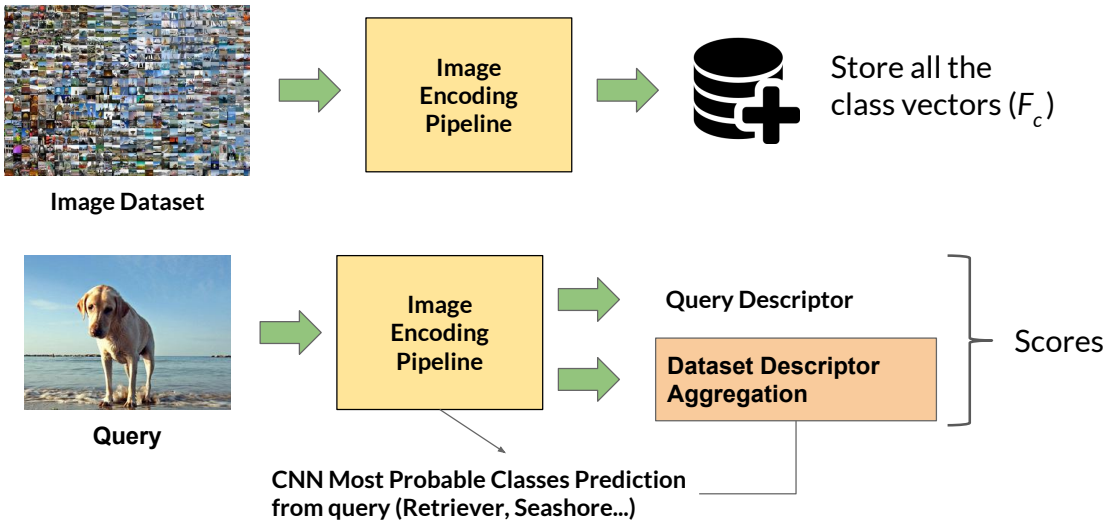


Figure 5.2: Online Aggregation Strategy. At test time we build the descriptors for all the dataset using the classes information of the query.

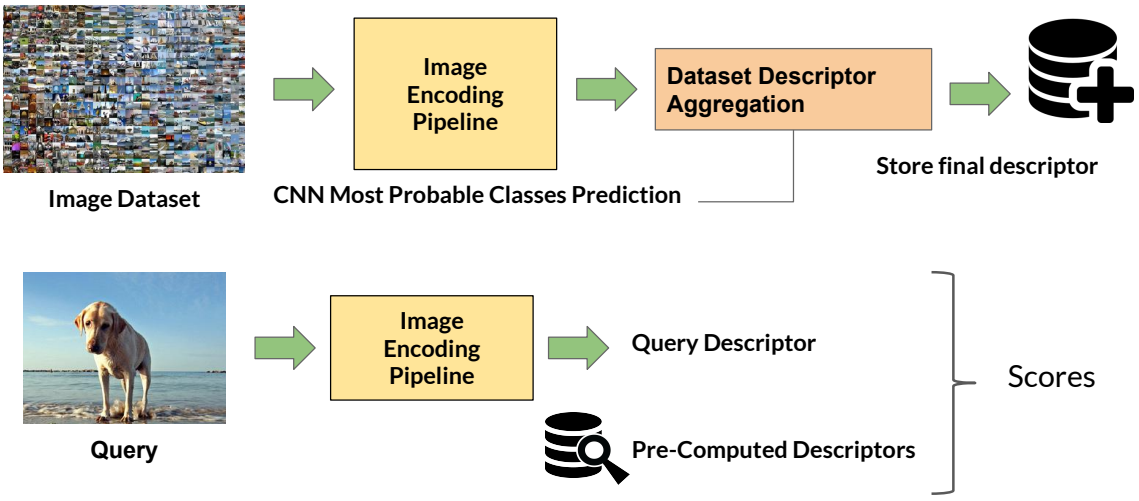


Figure 5.3: Offline Aggregation Strategy. The descriptors for all the dataset are computed offline. At test time we only compute the descriptor for the query.

Chapter 6

Experiments

In this Chapter we show and discuss the experiments we have done to validate our approach. In Section 6.1 we begin stating the experimental framework and datasets used. In Section 6.2 we provide a study of the technique proposed where we observe its behavior and sensitivity with respect to its parameters. In Sections 6.3 and 6.4 we compare our method with the state-of-the-art. Finally, in Section 6.5 we show some qualitative results.

6.1 Datasets and Experimental Details

We target the task of Visual Instance Search where, given an image query containing the object of interest, the search engine is expected to explore a large scale dataset to build a ranked list of images depicting the query object.

We present experiments in Oxford5k Buildings [19] and Paris6k Buildings [28]. Both datasets contain 55 query images to perform the search, each annotated with a region of interest. To test instance-level retrieval on a larger-scale scenario, we also consider the Oxford105k and the Paris106k datasets that extend Oxford5k and Paris6k with 100k distractor images collected from Flickr [19]. All the images in the datasets have a maximum size of 1024, so we keep it and resize the minimum dimension to 720, maintaining the aspect ratio. We follow the evaluation protocol using the features from the given query annotated region of interest. We compute the PCA parameters with Paris descriptors when we test in Oxford, and vice versa. We choose the cosine similarity metric to compute the scores as this operation is efficiently and fast computed with GPUs. The final ranked list is generated by ordering these scores. The evaluation metric for all the experiments is the mean Average Precision (mAP), as adopted in most related works using these datasets.

We explored the use of CAMs in different network architectures such as the recent DenseNet161 [31] and ResNet50 [32], the widely used VGG-16 [16] and DecomposeMe [34] which is a compact network based on 1D convolutions. In Figure 6.2 and in the appendices, we show some qualitative results showing the CAMs generated by these networks. We can observe that VGG-16 tends to focus more on the discriminative part of the objects objects rather than in their global shape, while being less spread around the image, a desirable property for our retrieval system.

To perform our experiments we use the CAM modified VGG-16 CNN [16] pre-trained on

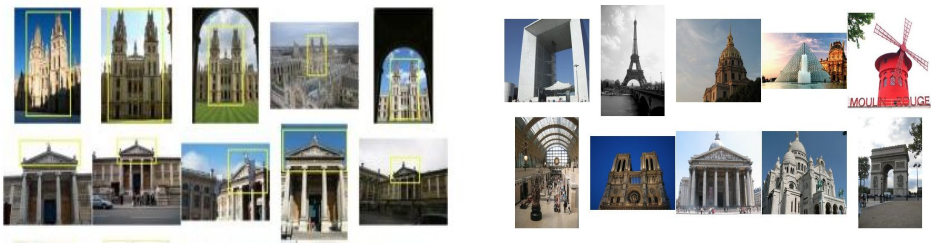


Figure 6.1: Example of images from Oxford5k (Left) and Paris6k (Right).

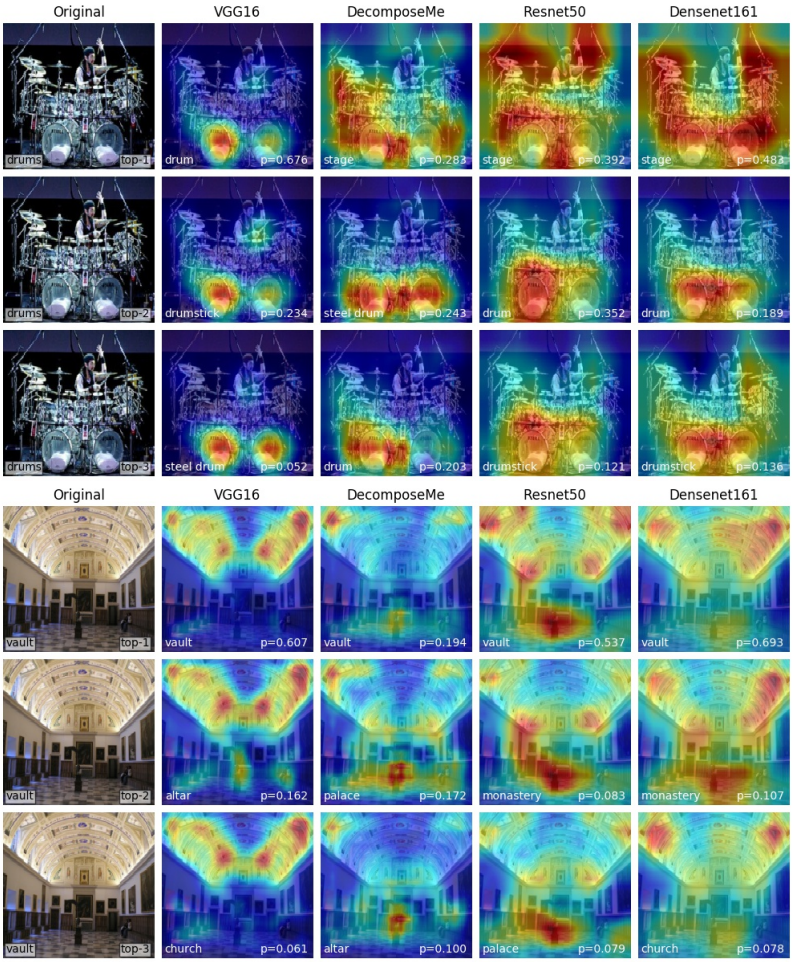


Figure 6.2: The first column correspond to the original image and the class ground truth, the second one to CAM VGG-16 version of [16], the third one to a CAM modified version of DecomposeMe [34], the fourth one to ResNet50 [32] and the last one to DenseNet-161 [31] (Both without any modification done). The rows correspond to the top-3 predicted classes for each image. In the bottom of each image the class predicted and the probability given by each network is shown.

the ILSVRC ImageNet dataset [13] as off-the-shelf model. Our experiments, shown in the next section, confirmed that provided the best performance of the three. Besides, this way, our results are comparable with the related work using pre-trained networks as well. In this modified VGG-16 [22] all the layers after the *conv5_3* layer have been removed, resulting on a final mapping

of 14×14 (for an input image of 224×224). On top of that, a convolutional layer of size 3×3, stride 1, zero-padding 1, with 1024 units, followed by a global average pooling layer and a linear layer with softmax normalization are added as shown in Figure 5.1. Features and CAMs extraction were implemented with the Keras[35] software framework with Theano [36] as backend for the VGG-16 model and PyTorch [37] for the rest of networks. We extract features from the last convolutional layers (*conv5_1*, *conv5_2*, *conv5_3*) and empirically determine the *conv5_1* as the one giving the best performance. As mentioned in [16], the CAM-modified model performs worse than the original VGG-16 in the task of classification, and we verify using a simple feature aggregation that the convolutional activations are worse for the retrieval case too. For Oxford dataset the relative differences are of 14.8% and 15.1% when performing max-pooling and sum-pooling, respectively. Although this drawback is later overcome by the benefits of the class mappings, it should be noted that applying our weights over the original VGG-16 convolutional features should improve even more the performance.

6.2 Ablation Studies

The model presented in Chapter 5 requires to tune two different parameters: the number of class vectors aggregated, N_C, and the number of classes we use to build the PCA matrix, N_{PCA}. The input matrix to compute the PCA has dimensions $N_{Im}N_{pca} \times K$ where N_{Im} and K are the number of images in the dataset and the number of feature maps of the convolutional layer considered, respectively.

The Online (OnA) and Offline (OfA) Aggregations are compared in Figure 6.3 in terms of mAP as a function of the amount of top N_C classes and N_{pca} classes used to compute the PCA. As a reference, the baseline mAP value obtained just sum-pooling the features and only adding CroW channel weights is shown in Table 6.1. Looking at these two experiments can observe that introducing the CAM spatial weighting is being beneficial and providing a huge improvement over the baseline results.

Descriptor Aggregation	Oxford5k	Paris6k
Raw Features	0.396	0.526
Raw + Crow (channel)	0.420	0.549
Raw Features + PCA	0.589	0.662
Raw + Crow(channel) + PCA	0.607	0.685

Table 6.1: Baseline results when computing the image descriptors by sum-pooling as in Equation 5.3 without using CAMs.

Next step is taking a look at our aggregation strategies proposed after including the CAM weighting (Figure 6.3). For the offline aggregation, the optimal N_C seems to be dataset dependent, Paris benefits from having more classes aggregated while the performance on Oxford dataset remains constant despite the number of classes. However, the patterns of online aggregation show that by aggregating few classes ($<$10) we are able to obtain a good performance for both datasets. Increasing the number of classes is also resulting in little benefit, mostly in Oxford dataset. It can be observed that knowing which content is relevant and building the descriptors accordingly results in a reduction of the class vectors required, as well as a performance boost. We observe that increasing the N_{pca} value does not improve the performance, suggesting that the randomness of the classes (of the target dataset) is not adding valuable information.

To improve the performance of the offline aggregation without the practical limitations of aggregating online, we suggest restricting the total number of classes used to the most probable classes of the dataset's theme. As we have two similar building datasets, Oxford and Paris, we compute the most representative classes of the 55 Paris queries and use that predefined list of classes ordered by probability of appearance to obtain the image representations in Oxford. The results can be observed in Figure 6.4. Firstly, we see that now we are learning a better PCA transformation when increasing N_{pca}. As we use the same classes per every image, PCA is finding a better representation space. Secondly, we see that the mAP improves for both OfA, as now we do not have the mismatching of classes, and OnA, because the PCA is providing a better transformation.

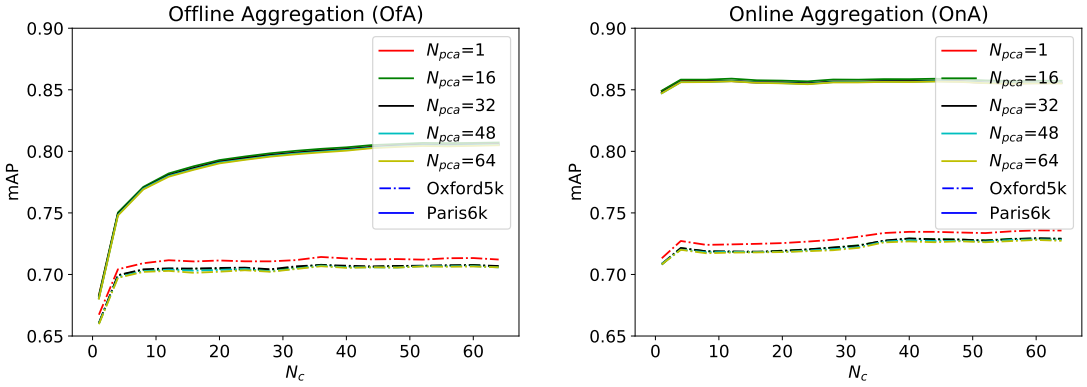


Figure 6.3: We show the sensitivity of our descriptor aggregation strategies with respect to the number of classes N_C used in the aggregation and the number of classes used to compute the PCA matrix N_{pca}. We report the mAP values for OfA (left) and OnA (Right). Straight line corresponds to Paris6k dataset while dashed corresponds to Oxford5k.

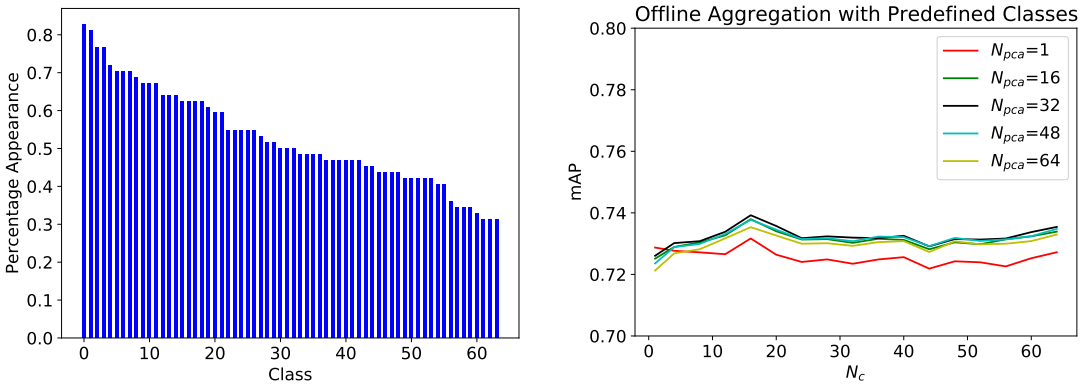


Figure 6.4: We show the appearance ratio of the selected classes in the 55 queries of Paris and the performance of the system when use a set of predefined classes and we vary N_C and N_{pca}. The first 16 classes correspond to: *vault, bell cote, bannister, analog clock, movie theater, coil, pier, dome, pedestal, flagpole, church, chime, suspension bridge, birdhouse, sundial, triumphal arch*.

In the next experiment we want to show the computational burden of adding the computation and application of CAMs in our pipeline for OfA strategy. Note that they are only computed for the top N_c classes. In Table 6.2 the computational overhead added can be seen for Oxford5k dataset. We can observe that using only 8 CAMs we obtain a substantial improvement just

adding $0.1s$ over the baseline time.

Descriptor Aggregation	Time (s)	mAP
Raw + PCA	0.49	0.589
1 CAM	0.5	0.667
8 CAMs	0.6	0.709
32 CAMs	0.9	0.711
64 CAMs	1.5	0.712

Table 6.2: Computational burden added by the use of CAMs

Finally, in Table 6.3 we provide a comparison of the different networks performance. The table shows the performance when computing the descriptors by sum-pooling the raw features and after the addition of CAMs. We observe that VGG-16 is the best of the three in all the cases. We hypothesize that the reason is that VGG is the one that it's looking at the most discriminative parts of the objects, rather than covering the whole object and some of the image content. The latter could be better for the classification task, but not for retrieval, where we are looking for particular characteristic details in images.

Network	Oxf5k	Paris6k
VGG-16 (Raw)	0.396	0.526
VGG-16 (64CAMs)	0.712	0.805
Resnet-50 (Raw)	0.389	0.508
Resnet-50 (64CAMs)	0.699	0.804
Densenet-161 (Raw)	0.339	0.495
Densenet-161 (64CAMs)	0.695	0.799

Table 6.3: Comparison of the retrieval performance employing different network architectures.

6.3 Comparison with the State-of-the-art

In this section we compare our proposal with with other state of the art works using the off-the-shelf VGG-16 network for image retrieval on the Oxford and Paris datasets. The comparison is shown in Table 6.4. As we have pointed before, OnA strategy is suitable for small datasets but it does not scale well for larger datasets. So we only use OfA mode with Oxf105k and Par106k that include the 100k distractors. The results that appear in the table are given for a N_{pca} of 1 and N_C of 64 for both approaches.

In Paris 6k benchmark we achieve the best result with our OnA strategy, with a notable difference with the OfA. This shows the importance of selecting the relevant image content. We can observe that our OfA method scales well, reaching the top performance in Oxford105k and falling behind RMAC [9] in Paris106k. If we are working in a particular application where we need to retrieve only specific content (e.g. buildings), the OfA strategy could be further enhanced by doing a filtering in the pool of possible classes as seen in Section 6.2.

Razavian et al. [8] achieve the highest performance in the Oxford benchmark by applying a extensive spatial search at different scales for all images in the database. However, the cost of their feature extraction is much higher than ours since they feed 32 image crops of resolution

576×576 to the CNN. In addition to have the descriptors that have the larger memory footprint. Our OnA proposal provides the third best result in the case of Oxford 5k, but the best in terms of descriptor dimension.

Method	Dim	Oxf5k	Par6k	Oxf105k	Par106k
SPoc [7]	256	0.531	-	0.501	-
uCroW [10]	256	0.666	0.767	0.629	0,695
Crow [10]	512	0.682	0.796	0.632	0.710
R-MAC [9]	512	0.669	0.830	0.616	**0.757**
BoW [24]	25k	0.738	0.820	0.593	0.648
Razavian [8]	32k	**0.843**	0.853	-	-
Ours(OnA)	512	0.736	**0.855**	-	-
Ours(OfA)	512	0.712	0.805	**0.672**	0.733

Table 6.4: Comparison with the state-of-the-art CNN based retrieval methods (Off-the-shelf). Dimensions of the descriptors are included in the second column (Dim).

6.4 Re-Ranking and Query Expansion

A common approach in image retrieval systems is to apply post-processing steps that refine a first fast search to improve the system performance. We have explored query expansion and re-ranking, as they are common choices in the related work [9, 10, 24].

Query Expansion: As we have seen in Section 2.3.1 there exist different ways to expand the query further than the image and a bounding box. We chose one of the simplest and fastest ones as in [10], by updating the query descriptor for the l_2 normalized sum of the top ranked QE descriptors.

Local-aware Re-Ranking As explained in Section 2.3.2 a first fast ranking based on the image features can be improved with a more detailed and local analysis over the top retrieved R images. This re-ranking is based on a more detailed matching between the query object and the location of this object in each top-R ranked images. There are multiple ways to obtain object locations. For instance, R-MAC [9] applies a fast spatial search with approximate max-pooling localization. BoW [24] applies re-ranking using a sliding window approach with variable bounding boxes. Our approach, in contrast, localizes objects on the images using class activation maps as explained in Chapter 4. We use the most probable classes predicted from the query to generate the regions of interest in the target images and with this information perform a spatial re-ranking after the first search. To obtain these regions, first we define heuristically a set of thresholds based on the normalized intensity of the CAM heatmap values. More precisely, we define a set of values 1%, 10%, 20%, 30% and 40% of the max value of the CAM and compute bounding boxes around the largest connected component of the CAM. Then, we build an image descriptor for every of the spatial regions and compare them to the query image using the cosine distance. We kept the one with the best score. The decision of using more than one threshold aims at covering the variability of the objects dimensions in different images and detect them with more precision. In Figure 6.5 we show an example of the regions of interest generated by this method. As in our datasets most of the objects of interest are composed by more of one class, using the average heatmap of the top-2 predicted classes improved a bit the bounding box generated and the final performance.

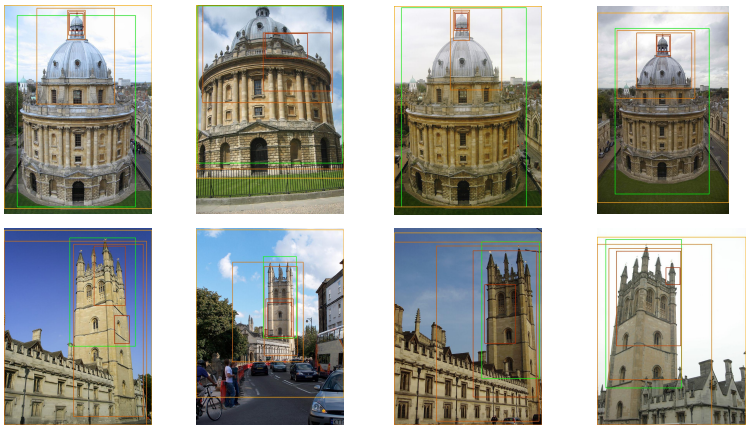


Figure 6.5: Some examples of regions of interest generated using CAMs. The green bounding box is the ground truth, while the rest from orange to red are the generated bounding boxes from the lower threshold to the highest one.

We provide a comparison of our re-ranking and query expansion results with the rest of the relevant literature. CroW [10] authors only apply query expansion after the initial search while BoW and R-MAC apply first a spatial re-ranking. The number of top-images used to employ these techniques varies between works. For the sake of comparison, Table 6.5 provide our results with the same parameters for query expansion (QE) and re-ranking (R). For the initial search we keep N_{pca} of 1 and N_C of 64 for both OnA and OfA as in the previous section, but for the re-ranking we decrease N_C to the 6 more probable classes. This is done because after the first search we already have a set of relevant images and what we want is to perform a more fine-grained comparison by looking at particular regions. In addition to that, taking less classes reduces the computational time spent.

Looking at Table 6.5, we can observe that our proposal achieves very competitive results even only with the application of a simple query expansion. If we add a re-ranking stage, we improve the performance mostly in Oxford dataset, where we obtain the top performance. In Paris, we can observe that performing the re-ranking step does not increase the performance mainly due to relevant images being already on the top QE of the ranked list.

Method	Dim	R	QE	Oxf5k	Par6k	Oxf105k	Par106k
CroW [10]	512	-	10	0.722	0.855	0.678	0.797
Ours(OnA)	512	-	10	0.760	0.873	-	-
Ours(OfA)	512	-	10	0.730	0.836	0.712	0.791
BoW [24]	25k	100	10	0.788	0.848	0.651	0.641
Ours(OnA)	512	100	10	0.780	0.874	-	-
Ours(OfA)	512	100	10	0.773	0.838	0.750	0.780
RMAC [9]	512	1000	5	0.770	**0.877**	0.726	**0.817**
Ours(OnA)	512	1000	5	**0.811**	0.874	-	-
Ours(OfA)	512	1000	5	0.801	0.855	**0.769**	0.800

Table 6.5: Comparison with the state-of-the-art after applying Re-Ranking (R) or/and Query Expansion (QE). Dimensions of the descriptors are included in the second column (Dim).

6.5 Qualitative Results

In this Section we illustrate and comment the search results for both aggregation strategies plus the results after adding query expansion and re-ranking. We have chosen one of the buildings that had the worse performance, the *Magdalen College* from Oxford5k. In the following figures we show the top-10 ranked images for 5 different queries of the building. There are 4 different labels of ground truth for the images. *Good* means that the object of interest is clearly visible (depicted with green border in the figures), *OK* means that more of 25% of the object is clearly visible (cyan border), *Junk* means that less of 25% of the object is visible (yellow border) and *Negative* that means that the object does not appear (red border).

In Figure 6.6 we show the offline aggregation results. We can observe that there are a lot of incorrect images retrieved, mainly because the region of interest consist of a small object located on the background.

Figure 6.6: Top-10 ranked results using Offline Aggregation for the *Magdalen College* building. The image in the blue border is the query, while good/ok/junk/negative images have green/-cyan/yellow/red border. In magenta is highlighted the query's annotated region of interest.

Figure 6.7 shows the results for Online Aggregation. Now the descriptors are built using the information from the query's classes. We can observe that there has been an improvement as the foreground classes (like the trees and gardens of the images), that would have been the most probable classes in the OfA strategy, are not interfering.

Figure 6.7: Top-10 ranked results using Online Aggregation for the *Magdalen College* building. The image in the blue border is the query, while good/ok/junk/negative images have green/-cyan/yellow/red border. In magenta is highlighted the query's annotated region of interest.

Finally, Figures 6.8 and 6.9 show the results for Online aggregation after applying query expansion with the top-5 ranked images and after re-ranking the top-1000 images. In the first one, we can observe that if the top images are not relevant, such as in the second and fourth rows, the application of query expansion is indeed detrimental for the performance of the system. While in the second case, we can observe that the re-ranking has brought very relevant images to the top ranked positions. Therefore, the expanded query is of better quality as can be seen in the qualitative results and in the performance tables.

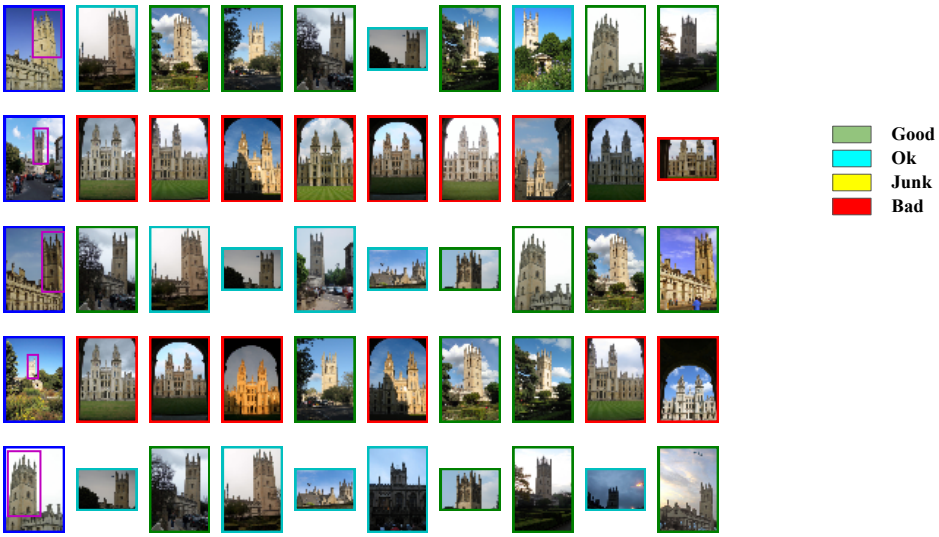


Figure 6.8: Top-10 ranked results using Online Aggregation plus Query Expansion ($QE = 10$) for the *Magdalen College* building. The image in the blue border is the query, while good/ok/junk/negative images have green/cyan/yellow/red border. In magenta is highlighted the query's annotated region of interest.

Figure 6.9: Top-10 ranked results using Online Aggregation plus Re-Ranking ($R = 1000$) and Query Expansion ($QE = 5$) for the *Magdalen College* building. The image in the blue border is the query, while good/ok/junk/negative images have green/cyan/yellow/red border. In magenta is highlighted the query's annotated region of interest.

Chapter 7

Conclusions

In this thesis we focused on the task of content-based image retrieval and, in particular, on the visual instance search case, where the objective is to search for particular objects in a dataset of images.

Our proposal is based on codifying images dynamically looking at their semantic content using only the knowledge contained inside the network. To this end, our technique includes an image encoding pipeline that makes use of a pre-trained CNN and Class Activation Maps to extract discriminative regions from the image and then weight its convolutional features accordingly.

We presented a comprehensive set of experiments in publicly available datasets. In a first experiment, we focus on an ablation study where we test our technique tuning its parameters. We proposed a way of improving the performance of our algorithm by fixing predetermined classes when targeting a specific dataset theme. From our experimental results, we can conclude that locating the most discriminative parts in the objects is more beneficial that focusing on the whole object and context.

In a second set of experiments, we demonstrate that adding CAM weighting provides a great performance boost compared to the baseline. In addition, adding this weighting process does not imply a large overhead in computation time. From our experimental results, we can conclude that selecting the relevant content of the image to build its descriptor is helpful and contributes to improve the overall performance. Our approach establishes a new state-of-the-art in publicly available datasets outperforming other methods that build image representations combining off-the-shelf features focusing on random or fixed grid regions.

This work was submitted in the form of a paper and accepted in the BMVC2017 Conference to be held in London in September 2017. This is the fourth most important conference in computer vision, with an h5-index of 41 and h5-median of 72 (Google Scholar, checked on 4th July 2017). Its acceptance rate in 2016 was 39%.

Bibliography

[1] Google. Search by image. `https://www.google.com/intl/es419/insidesearch/features/images/searchbyimage.html`.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[4] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[5] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014.

[6] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014.

[7] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, 2015.

[8] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. A baseline for visual instance retrieval with deep convolutional networks. *arXiv preprint arXiv:1412.6574*, 2014.

[9] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *ICLR*, 2016.

[10] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *ECCV*. Springer, 2016.

[11] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *ECCV*, 2016.

[12] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. *arXiv preprint arXiv:1604.02426*, 2016.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[14] Shiliang Zhang, Ming Yang, Xiaoyu Wang, Yuanqing Lin, and Qi Tian. Semantic-aware co-indexing for image retrieval. In *ICCV*, 2013.

[15] X Yu Felix, Rongrong Ji, Ming-Hen Tsai, Guangnan Ye, and Shih-Fu Chang. Weak attributes for large-scale image retrieval. In *CVPR*, 2012.

[16] Bolei Zhou, Aditya Khosla, Lapedriza. Agata, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[17] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[18] Josef Sivic, Andrew Zisserman, et al. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[19] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[20] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[21] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, 2014.

[24] Eva Mohedano, Kevin McGuinness, Noel E O'Connor, Amaia Salvador, Ferran Marqués, and Xavier Giró-i Nieto. Bags of local convolutional features for scalable instance search. In *ICMR*, 2016.

[25] Amaia Salvador, Xavier Giró-i Nieto, Ferran Marqués, and Shin'ichi Satoh. Faster r-cnn features for instance search. In *CVPR Workshop*, 2016.

[26] Cristian Reyes, Eva Mohedano, Kevin McGuinness, Noel E O'Connor, and Xavier Giro-i Nieto. Where is my phone?: Personal object retrieval from egocentric images. In *LTA Workshop*, 2016.

[27] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, 2007.

[28] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[30] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition. `http://cs231n.github.io/convolutional-networks/`.

[31] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[33] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *ECCV*. 2012.

[34] Jose Alvarez and Lars Petersson. Decomposeme: Simplifying convnets for end-to-end learning. *arXiv preprint arXiv:1606.05426*, 2016.

[35] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

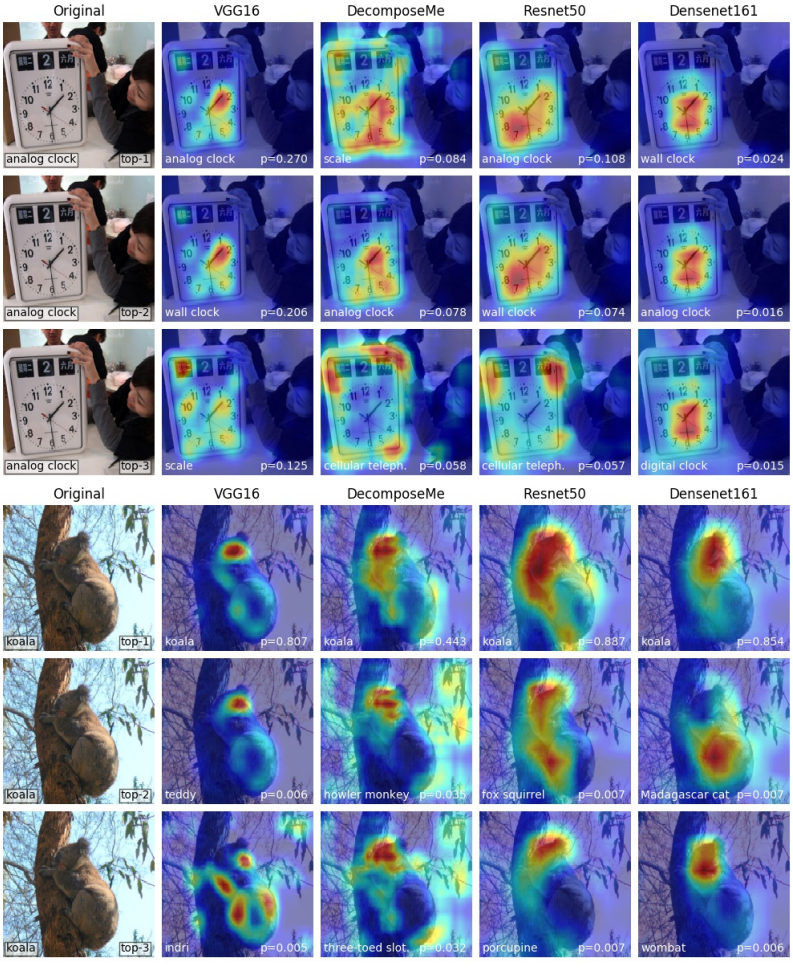
[36] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, 2016.

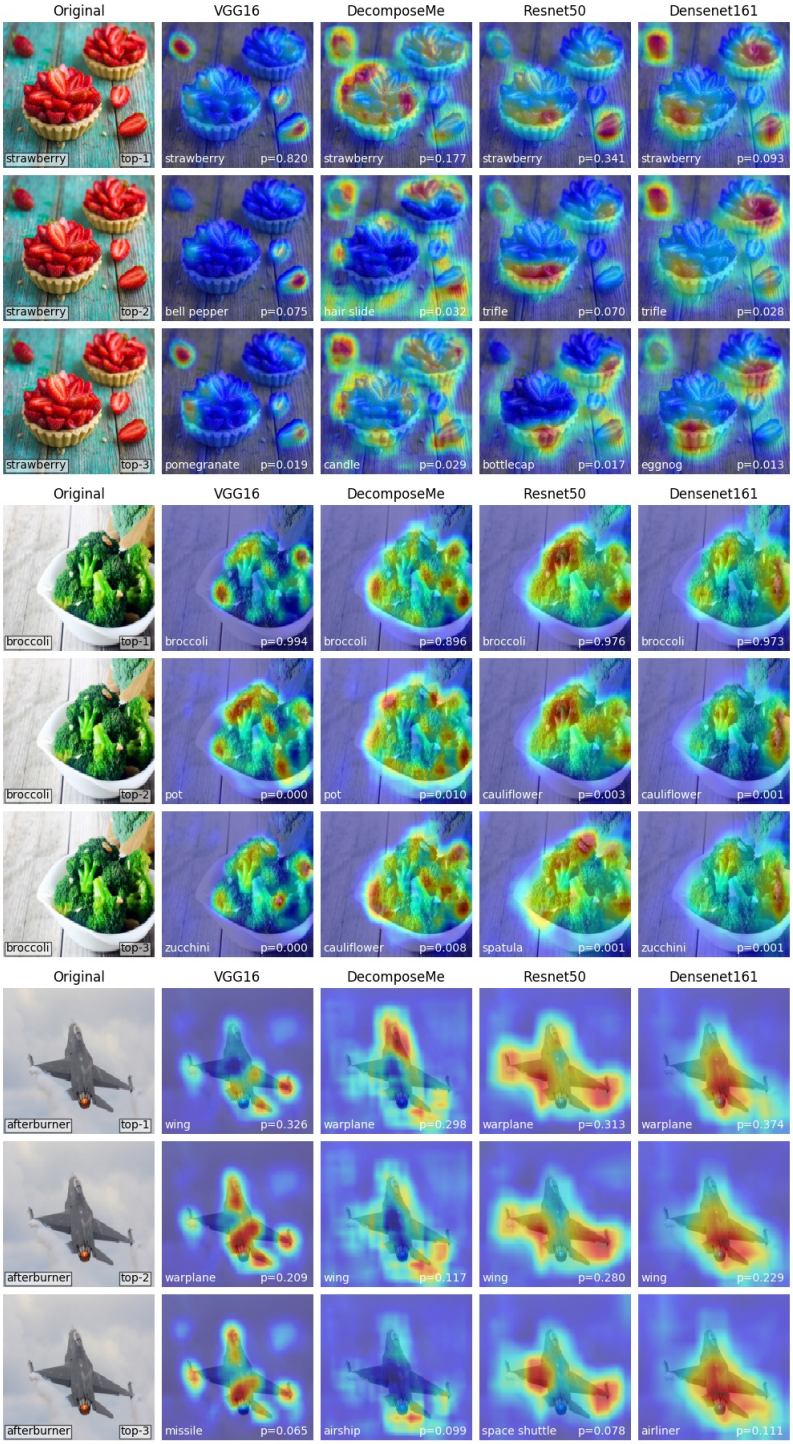
[37] Soumith Chintala, Adam Paszke, and Sam Gross. Pytorch. `https://github.com/pytorch/pytorch`, 2016.

Appendix A

CAMs Examples

This appendix includes qualitative examples of CAMs using different network architectures. The first column correspond to the original image and the class ground truth, the second one to CAM VGG-16 version of [16], the third one to a CAM modified version of DecomposeMe [34], the fourth one to ResNet50 [32] and the last one to DenseNet-161 [31] (Both without any modification done). The rows correspond to the top-3 predicted classes for each image. In the bottom of each image the class predicted and the probability given by each network is shown.




Original
VGG16
DecomposeMe
Resnet50
Densenet161
strawberry top-1
strawberry p=0.820
strawberry p=0.177
strawberry p=0.341
strawberry p=0.093
strawberry top-2
bell pepper p=0.075
hair slide p=0.032
trifle p=0.070
trifle p=0.028
strawberry top-3
pomegranate p=0.019
candle p=0.029
bottlecap p=0.017
eggnog p=0.013
Original
VGG16
DecomposeMe
Resnet50
Densenet161
broccoli top-1
broccoli p=0.994
broccoli p=0.896
broccoli p=0.976
broccoli p=0.973
broccoli top-2
pot p=0.000
pot p=0.010
cauliflower p=0.003
cauliflower p=0.001
broccoli top-3
zucchini p=0.000
cauliflower p=0.008
spatula p=0.001
zucchini p=0.001
Original
VGG16
DecomposeMe
Resnet50
Densenet161
afterburner top-1
wing p=0.326
warplane p=0.298
warplane p=0.313
warplane p=0.374
afterburner top-2
warplane p=0.209
wing p=0.117
wing p=0.280
wing p=0.229
afterburner top-3
missile p=0.065
airship p=0.099
space shuttle p=0.078
airliner p=0.111

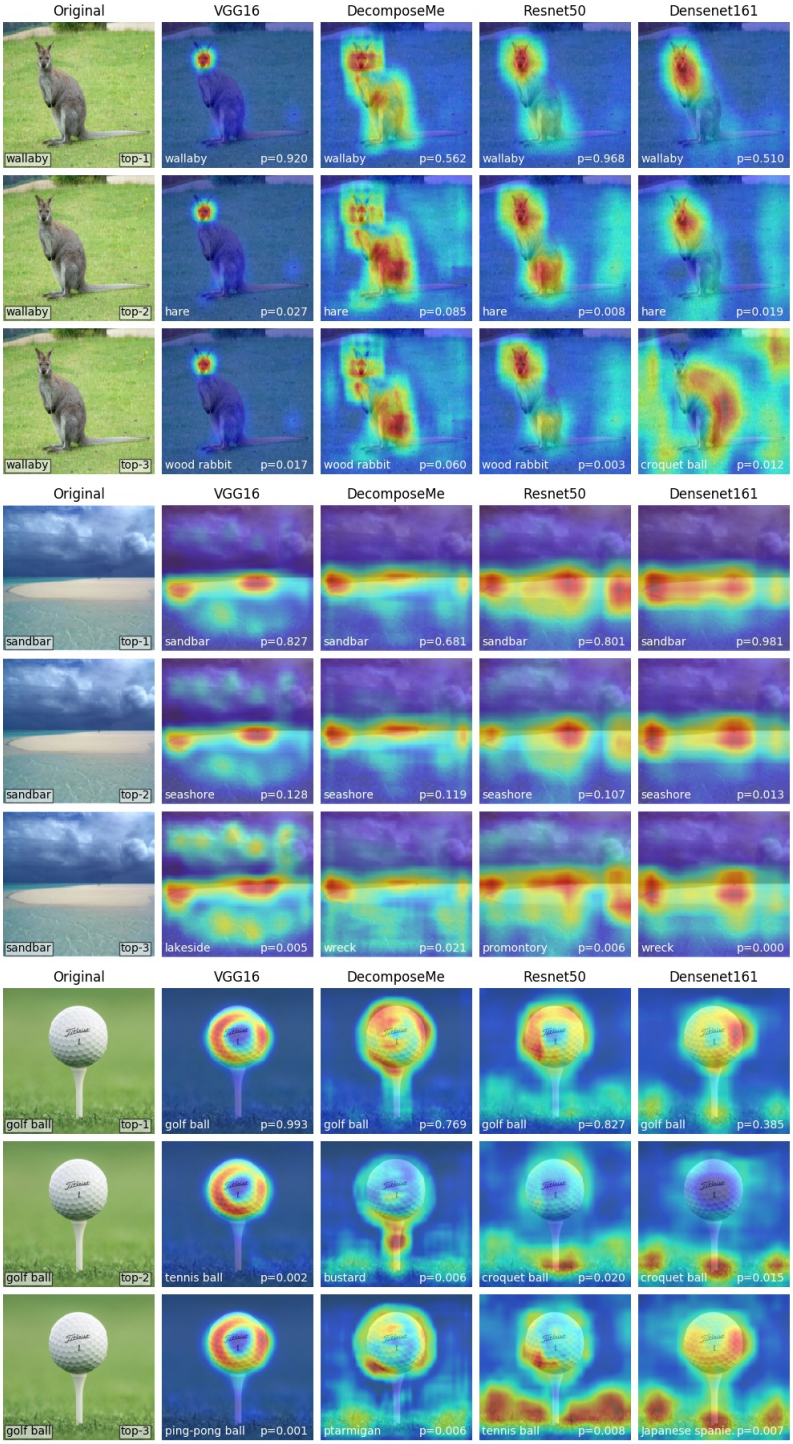


Figure A.1: The first column correspond to the original image and the class ground truth, the second one to CAM VGG-16 version of [16], the third one to a CAM modified version of DecomposeMe [34], the fourth one to ResNet50 [32] and the last one to DenseNet-161 [31] (Both without any modification done). The rows correspond to the top-3 predicted classes for each image. In the bottom of each image the class predicted and the probability given by each network is shown.

Appendix B

Retrieval Examples

This appendix includes another example of search results for both aggregation strategies plus the results after adding query expansion and re-ranking. In the following figures we show the top-10 ranked images for 5 different queries of the building *Cornmarket* of OXford5k. There are 4 different labels of ground truth for the images. *Good* means that the object of interest is clearly visible (depicted with green border in the figures), *OK* means that more of 25% of the object is clearly visible (cyan border), *Junk* means that less of 25% of the object is visible (yellow border) and *Negative* that means that the object does not appear (red border).

Figure B.1: Top-10 ranked results using Offline Aggregation for the *Cornmarket* building.

Figure B.2: Top-10 ranked results using Online Aggregation for the *Cornmarket* building.

Figure B.3: Top-10 ranked results using Online Aggregation and QE (top-10) for the *Cornmarket* building.

Figure B.4: Top-10 ranked results using Online Aggregation, Re-Ranking (top-1000) and QE (top-5) for the *Cornmarket* building.

Appendix C

BMVC2017 Paper

This appendix includes the paper accepted for presentation at British Machine Vision Conference (BMVC) in London on the 4th-7th September 2017.

Class Weighted Convolutional Features for Visual Instance Search

Albert Jimenez[1]
albertjimenezsanfiz@gmail.com
Jose M. Alvarez[2]
http://www.josemalvarez.net
Xavi Giro-i-Nieto[1]
xavier.giro@upc.edu

[1] Image Processing Group
Universitat Politecnica de Catalunya
Barcelona, Spain

[2] CSIRO / Data61
Canberra, Australia

Abstract

Image retrieval in realistic scenarios targets large dynamic datasets of unlabeled images. In these cases, training or fine-tuning a model every time new images are added to the database is neither efficient nor scalable. Convolutional neural networks trained for image classification over large datasets have been proven effective feature extractors when transferred to the task of image retrieval. The most successful approaches are based on encoding the activations of convolutional layers, as they convey the image spatial information. Our proposal goes beyond and aims at a local-aware encoding of these features depending on the predicted image semantics, with the advantage of using only of the knowledge contained inside the network. In particular, we employ Class Activation Maps (CAMs) to obtain the most discriminative regions of the image from a semantic perspective. Additionally, CAMs are also used to generate object proposals during an unsupervised re-ranking stage after a first fast search. Our experiments on two public available datasets for instance retrieval, Oxford5k and Paris6k, demonstrate that our system is competitive and even outperforms the current state-of-the-art when using off-the-shelf models trained on the object classes of ImageNet.

1 Introduction

Content-based Image Retrieval (CBIR) and, in particular, object retrieval (instance search) is a very active field in computer vision. Given an image containing the object of interest (visual query), a search engine is expected to explore a large dataset to build a ranked list of images depicting the query object. This task has been addressed in multiple ways: from learning efficient representations [17, 20] and smart codebooks [2, 18], to refining a first set of quick and approximate results with query expansion [7, 13, 30] or spatial verification [18, 27].

Convolutional neural networks trained on large scale datasets have the ability of transferring the learned knowledge from one dataset to another [32]. This property is specially important for the image retrieval problem, where the classic study case targets a large and growing dataset of unlabeled images. Therefore, approaches where a CNN is re-trained every time new images are added does not scale well in a practical situation.

Many works in the literature focus on using a pre-trained CNNs as feature extractor or, alternatively, using enhanced features by performing a fine-tuning step on a custom dataset. For instance, [4] and [9] proposed the use of the activations of the fully-connected layers. Other later works demonstrated that the activations of convolutional layers provide better performance, as they convey the spatial information which is relevant for object retrieval [3]. Following this observation, other works based their approach on combining convolutional features with regions of interest inside the image [3, 15, 22, 31]. More recent works have focused on applying supervised learning to fine-tune CNNs using similarity oriented loss functions such as ranking [10] or pairwise similarity [21]. Adapting the CNN boosts the performance of the representations obtained. However, it has the main drawback of having to spend large efforts on collecting, annotating and cleaning a large dataset, which sometimes is not feasible.

In our work we aim at building better image representations by making use of the image semantics. That includes extracting information from two different layers of a pre-trained network: from a convolutional one and the last fully connected one. Besides, we are not training or fine-tuning a model specifically for the retrieval task or for a particular dataset, we are just taking profit of the built-in network knowledge. We base our argument in the fact that features learned from a large scale dataset [24] can be transferable to any other datasets [32]. The proposed solution can be directly applied to a broad domain of datasets with natural images, with no need of adaptation. The key idea of our approach is obtaining improved image representations by explicitly encoding the spatial information about the objects in the image. There are works like [33] that use both semantic attributes and local features to compute inverted indexes for fast retrieval, or [8] that propose to use an embedding of weak semantic attributes. However, most of the existing retrieval techniques omit computing regions of the image associated with the objects that appear inside, mainly because it relies in other expensive approaches like training an object detector. Our proposal makes use of Class Activation Maps (CAMs) [34], allowing us to leverage the semantic information contained in a pre-trained CNN in an elegant fashion.

The main contributions of this paper are: First, we propose to encode images based on their semantic information by using CAMs to spatially weight convolutional features. Second, we propose to use the object mappings given by CAMs to compute fast regions of interest for a posterior re-ranking stage. Finally, we set a new state-off-the art in Oxford5k and Paris6k using off-the-shelf features.

2 Related Work

Following the success of CNNs for the task of image classification, recent retrieval works have replaced the classical hand-crafted features for representations extracted from off-the-shelf CNNs. A first approach was using features extracted from the fully-connected layers of the networks, called Neural Codes [4]. An extension to local analysis was presented in [26], where these features were extracted over a fixed set of regions at different scales defined over the image.

Later, it was observed that features from convolutional layers convey the spatial information of images making them more useful for the task of retrieval. Based on this observation, different authors have based their approaches on combining convolutional features with different estimation of the areas of interest within the image. R-MAC [31] and BoW [16] use a fixed rigid grid of regions, [26] considers random regions, SPoC [3] assumes that the relevant

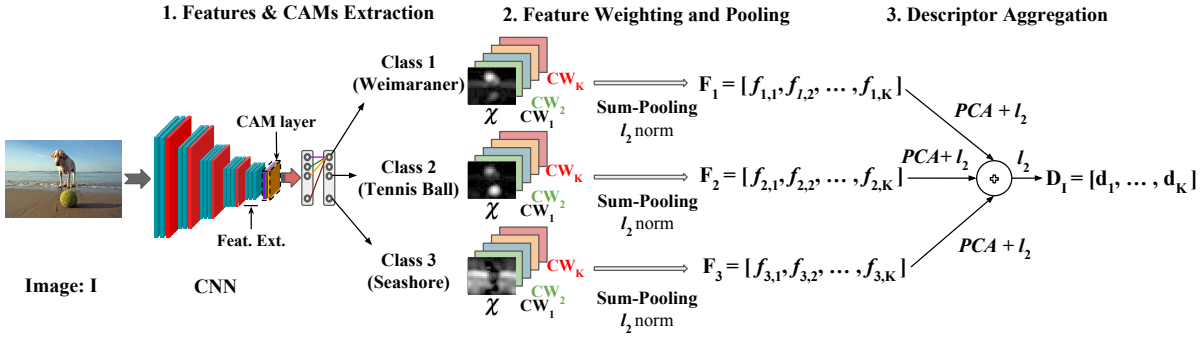


Figure 1: Image encoding into a compact descriptor. After forwarding the image, its feature maps are weighted spatially by each CAM, sum-pooled and weighted by channel. Then, PCA and l_2 normalization are applied to each class vector F_c. Finally, they are aggregated and l_2 normalized again to build a compact representation D.

content is in the center of the image (dataset bias) and [26] considers random regions. These works show how focusing on local regions of the convolutional features maps is improving the performance, but the computation of these regions is based on heuristics and randomness, not on the image content.

In this work, we aim at extracting features with focus on local areas that depend on the contents of the image, as other authors have explored in the past. For instance, in [10, 25], a region proposal network is trained for each query object. However, this solution does not scale well as it is a computational intensive process that must be run at query time, both for the training, and for the analysis of a large scale dataset at search time. Other approaches use an additional model to predict regions of interest for each image. For example, the work in [23] uses saliency maps generated by an eye gaze predictor to weight the convolutional features. However, this option requires additional computation of the saliency maps and therefore duplicates the computational effort of indexing the database. Yet another approach is proposed by the CroW model [15]. This model estimates a spatial weighting of the features as a combination of convolutional feature maps across all channels of the layer. As a result, features at locations with salient visual content are boosted while weights in non-salient locations are decreased. This weighting scheme can be efficiently computed in a single forward pass. However, it does not explicitly leverage semantic information contained in the model. In the next section, we present our approach based on Class Activation Maps [34] to exploit the predicted classes and obtain semantic-aware spatial weights for convolutional features.

3 Class-Weighted Convolutional Features

We aim at encoding images into compact representations, taking into account the semantics of the scene and using only the built-in network knowledge. To achieve that, we use convolutional features weighted by a soft attention model over the classes contained in the image. Our main contribution is exploiting the transferability of the information encoded in a CNN, not only in its features, but also in its ability to focus the attention on the most representative regions of the image. For this goal, we explore the potential of Class Activation Maps (CAMs) [34] to generate sematic-aware weights for convolutional features extracted from the deeper layers of a network.

In Section 3.1 we provide a review of the CAMs framework. Figure 1 presents the whole pipeline of our proposal, which is described in detail in Section 3.2.

3.1 Class Activation Maps

Class Activation Maps (CAMs) [34] were proposed as a method to estimate the pixels of the image that were most attended by the CNN when predicting each semantic class. The computation of CAMs is straightforward in most state-of-the-art CNN architectures for image classification by replacing the last fully-connected layers with a Global Average Pooling (GAP) layer and a linear layer. In the original work they add another convolutional layer before the GAP (*CAM layer*) to improve the performance lost after the removal of the fully-connected ones. In the case of having a network architecture where the layer before the classifier is a GAP layer, CAMs can be directly extracted without any kind of network modification. The details of our specific architecture can be seen in Section 4.1.

Given an output semantic class c, its CAM is computed as a linear combination of the feature maps in the last convolutional layer, weighted by the class weights learned by the linear classifier. More precisely, the computation of the CAM for the $c-th$ class, CAM_c, is as follows:

$$CAM_c = \sum_{k=1}^{K} conv_k \cdot w_{k,c} \tag{1}$$

where $conv_k$ is the kth feature map of the last convolutional layer before the GAP layer, and $w_{k,c}$ is the weight associated with that feature map k and class c. Notice that, as we are applying a global average pooling before the classifier, the CAM architecture is valid for images of any size.

From CAMs is possible to extract bounding boxes estimating the localization of objects [34]. The procedure consists of setting a threshold based on the normalized intensity of the CAM heatmap values and then setting to zero all the values below that threshold. Then define as a region of interest the bounding box that covers the largest connected element. In section 4.4 we explain how we have adapted this algorithm for our particular use.

3.2 Image Encoding Pipeline

The image encoding pipeline depicted in Figure 1 is divided in three stages:

1. Convolutional Features and CAMs Extraction. Each image is feed-forwarded through the CNN to compute, in a single pass, the convolutional features of a selected layer and the CAMs. The selected convolutional layer has K feature maps (χ) of width W and height H. Every CAM highlights the class-specific discriminative regions attended by the network to make its predictions. CAMs are normalized to fall in the range $[0,1]$ and, if their dimensions do not match the ones of the selected convolutional feature maps, they must be finally resized.

2. Feature Weighting and Pooling. Once the convolutional features and the CAMs have been extracted, the next step is weighting the features and pooling them to obtain a compact representation. For a given class c, we weight its features spatially, multiplying element-wise by the corresponding normalized CAM. Afterwards, each convolutional feature map is reduced to a single value by sum-pooling, in such a way that the dimensions of the feature vector corresponds to the amount of convolutional filters in the selected layer. We choose sum-pooling instead of max-pooling because we want to cover the extension of the objects rather than the most discriminative part. Furthermore, sum-pooling aggregation benefits more from the later application of PCA and whitening, as we observed experimentally and equally noted in [3, 15]. Finally, we include the channel weighting proposed in CroW [15] to reduce channel redundancies and augment the contribution of rare features. More precisely,

the channel weighting CW_k is computed as the logarithm of the inverse channel sparsity [15]:

$$CW_k = \log\left(\frac{\sum_{n=1}^{K}\left(\frac{\sum_{i,j} 1\ [\chi_{i,j}^{(n)}>0]}{WH}\right)}{\frac{\sum_{i,j} 1\ [\chi_{i,j}^{(k)}>0]}{WH}}\right). \tag{2}$$

Finally, the class vector is obtained as $F^c = [f_1^c, f_2^c, ..., f_K^c]$, where K is the depth of convolutional layer. Each of its components is computed as follows,

$$f_k^{(c)} = CW_k \sum_{i=1}^{W}\sum_{j=1}^{H} \chi_{i,j}^{(k)}\ CAM_{i,j}^{(c)}. \tag{3}$$

3. Aggregation of Semantic-Aware Descriptors. The final step is building a descriptor D_I for each image I by aggregating N_C class vectors. We perform l_2 normalization, PCA-whitening [14] and l_2 normalization once more as in [15, 31]. Then we combine the number of class vectors into a single one by summing them and l_2 normalizing again in the end.

All the classes that we have available to aggregate are given by a pre-trained CNN. As we are transferring the learning into other datasets, we have to define a policy to select which classes are the most relevant. We define two basic approaches depending on the moment when we build the descriptors for the dataset:

Online Aggregation (OnA). The top N_C predicted classes of the query image are obtained at search time (online) and the same set of classes is used to aggregate the features of each image in the dataset. This strategy, while generating descriptors that adapt to the query, presents two important drawbacks which do not make it scalable. First, it requires extracting and storing the CAMs for all classes for every image from the target dataset, with the corresponding requirements in terms of computation and storage. Secondly, the aggregation of weighted feature maps must also be computed at query time, which slows down the retrieval process.

Offline Aggregation (OfA). The considered top N_C semantic classes can also be predicted individually for each image in the dataset at indexing time. This task is performed offline and no intermediate information needs to be stored, just the final descriptor, which makes the system more scalable than the online approach.

4 Experiments

4.1 Datasets and Experimental Details

We present experiments in Oxford 5k Buildings [18] and Paris 6k Buildings [19]. Both datasets contain 55 query images to perform the search, each annotated with a region of interest. To test instance-level retrieval on a larger-scale scenario, we also consider the Oxford 105k and the Paris 106k datasets that extend Oxford 5k and Paris 6k with 100k distractor images from collected from Flickr [18]. All the images in the datasets have a maximum size of 1024, so we keep it and resize the minimum dimension to 720, maintaining the aspect ratio. We follow the evaluation protocol using the convolutional features of the query's annotated region of interest. We compute the PCA parameters in Paris when we test in Oxford, and vice versa. We choose the cosine similarity metric to compute the scores as this operation is efficiently and fast computed with GPUs. The evaluation metric for all the experiments is the mean Average Precision (mAP).

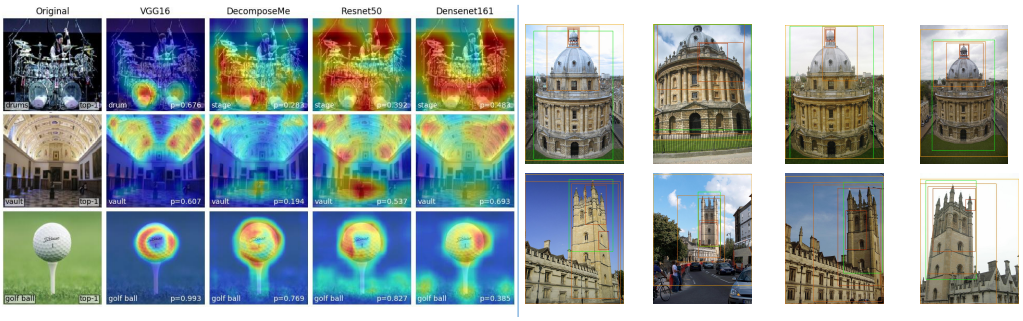


Figure 2: Left: Qualitative examples of CAMs using different architectures. The first column corresponds to (VGG-16-CAM [34]), the second one to a CAM modified version of DecomposeMe [1], the third and fourth ones to ResNet-50 [11] and DenseNet-161 [12], respectively. The rows correspond to the predicted image semantic classes and include the probability assigned. Right: Some examples of regions of interest generated using CAMs. The green bounding box is the ground truth, while the rest from orange to red are the generated bounding boxes from the lowest threshold to the highest one.

We explored the use of CAMs in different network architectures such as the recents DenseNet-161 [12], ResNet-50 [11], DecomposeMe [1] which is a compact network based on 1D convolutions and the widely used VGG-16 [28] after the CAM modification[34]. In Figure 2 we show some qualitative results where we show that VGG-16 tends to focus more on particular objects rather than in the global context of the image which is a desirable property for our system as well as having a final descriptor length of 512, which is four times less than ResNet-50. To perform the experiments we use the modified model proposed by [34] pre-trained on the ILSVRC ImageNet dataset [24] as our previous experiments confirmed that performed the best of the three (Table 1), this way, our results are also comparable with the related work using pre-trained networks as well. Features and CAMs extraction was implemented with the Keras [5] software framework with Theano [29] as backend. We extract features from the last convolutional layers (*conv5_1*, *conv5_2*, *conv5_3*) and empirically determine the *conv5_1* as the one giving the best performance. As mentioned in [34], the CAM-modified model performs worse than the original VGG-16 in the task of classification, and we verify using a simple feature aggregation that the convolutional activations are worse for the retrieval case too. For Oxford dataset the relative differences are of 14.8% and 15.1% when performing max-pooling and sum-pooling, respectively.

4.2 Ablation Studies

The model presented in Section 3.2 requires two different parameters to tune: the number of class vectors aggregated N_C, and the number of classes used to build the PCA matrix, N_{PCA}. The input matrix to compute it has dimensions $N_{Im}N_{pca} \times K$ where N_{Im} and K are the number of images in the dataset and the number of feature maps of the convolutional layer considered, respectively.

The *Online (OnA)* and *Offline (OfA) Aggregations* are compared in Figure 3 in terms of mAP as a function of the amount of top N_C classes and N_{pca} classes used to compute the PCA. As a reference, the **baseline mAP** values obtained just sum-pooling the features and applying PCA correspond to **0.589** for Oxford5k and **0.6626** for Paris6k and if we add the channel weighting they increase until **0.608** and **0.686** respectively. Our technique improves the baseline without adding a large computational overhead as can be seen in Table 2

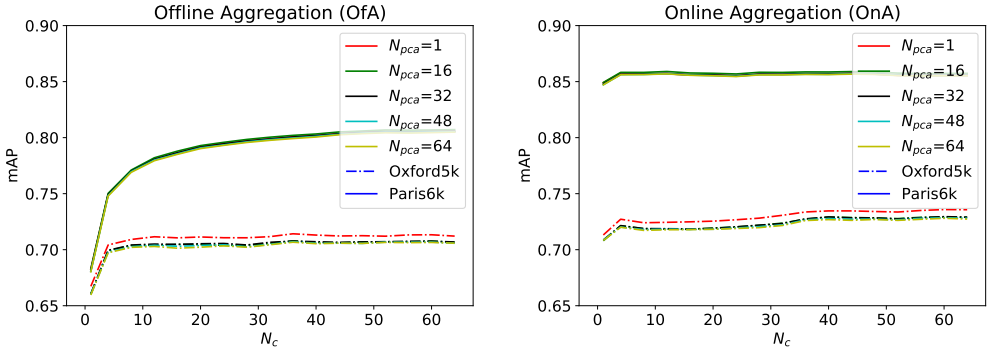


Figure 3: We show the sensitivity of our descriptor aggregation strategies with respect to the number of classes N_C used in the aggregation and the number of classes used to compute the PCA matrix N_{pca}. We report the mAP values for OfA (left) and OnA (Right). Straight line corresponds to Paris6k dataset while dashed corresponds to Oxford5k.

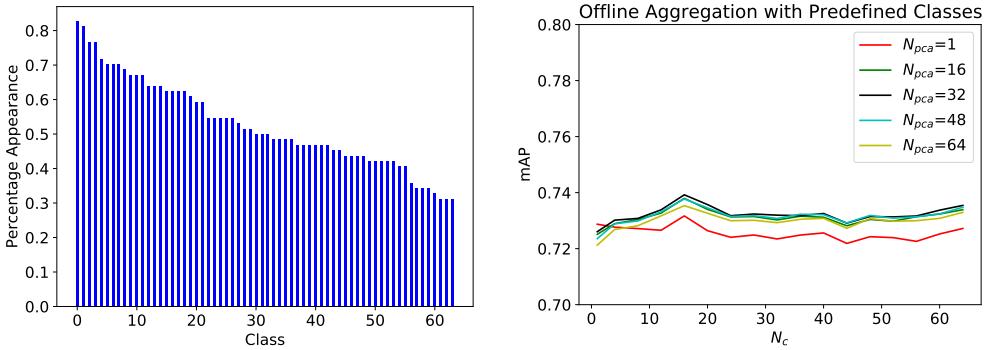


Figure 4: We show he appearance ratio of the selected classes in the 55 queries of Paris and the performance of the system when use a set of predefined classes and we vary N_C and N_{pca}. The first 16 classes correspond to: *vault, bell cote, bannister, analog clock, movie theater, coil, pier, dome, pedestal, flagpole, church, chime, suspension bridge, birdhouse, sundial, triumphal arch.*

For the offline aggregation, the optimal N_C seems to be dataset dependent, Paris benefits from having more classes aggregated while the performance on Oxford dataset remains constant despite the number of classes. However, the patterns of online aggregation show that aggregating few classes (< 10) we are able to obtain a good performance for both datasets. Increasing the number of classes is also resulting in little benefit, mostly in Oxford dataset. It can be observed that knowing the which content is relevant and building the descriptors results accordingly in a reduction of the class vectors required, as well as a performance boost. We observe that increasing the N_{pca} value does not improve the performance, suggesting that the randomness of the classes (of the target dataset) is not adding valuable information.

To improve the performance of the offline aggregation without the practical limitations of aggregating online, we suggest restricting the total number of classes used to the most probable classes of the dataset's theme. As we have two similar building datasets, Oxford and Paris, we compute the most representative classes of the 55 Paris queries and use that predefined list of classes ordered by probability of appearance to obtain the image representations in Oxford. The results can be observed in Figure 4. Firstly, we see that now we are learning a better PCA transformation when increasing N_{pca}. As we use the same classes per every image, PCA is finding a better representation space. Secondly, we see that the mAP improves for both OfA, as now we do not have the mismatching of classes, and OnA, because the PCA is providing a better transformation.

Method	Dim	Oxf5k	Par6k	Oxf105k	Par106k
SPoC [3]	256	0.531	-	0.501	-
uCroW [15]	256	0.666	0.767	0.629	0,695
CroW [15]	512	0.682	0.796	0.632	0.710
R-MAC [31]	512	0.669	0.830	0.616	**0.757**
BoW [16]	25k	0.738	0.820	0.593	0.648
Razavian [22]	32k	**0.843**	0.853	-	-
Ours(OnA)	512	0.736	**0.855**	-	-
Ours(OfA)	512	0.712	0.805	**0.672**	0.733

(a)

Network	Oxf5k	Paris6k
VGG-16 (Raw)	0.396	0.526
VGG-16 (64CAMs)	0.712	0.805
Resnet-50 (Raw)	0.389	0.508
Resnet-50 (64CAMs)	0.699	0.804
Densenet-161 (Raw)	0.339	0.495
Densenet-161 (64CAMs)	0.695	0.799

(b)

Table 1: (a) Comparison with the state-of-the-art CNN based retrieval methods (Off-the-shelf). (b) Performance comparison using different networks.

4.3 Comparison with the State-of-the-art

Table 1 compares our proposal with other state-of-the-art works that are using VGG-16 network off-the-shelf for image retrieval on the Oxford and Paris datasets. As we have pointed before, OnA is suitable for small datasets but it does not scale well for larger datasets. So we only use OfA mode with Oxf105k and Par106k that include the 100k distractors. The results that appear in the table are given for a N_{pca} of 1 and N_C of 64 for both approaches.

In Paris 6k benchmark we achieve the best result with our OnA strategy, with a notable difference with the OfA. This shows the importance of selecting the relevant image content. We can observe that our OfA method scales well, reaching the top performance in Oxford105k and falling behind RMAC [31] in Paris106k. If we are working in a particular application where we need to retrieve only specific content (e.g. buildings), the OfA strategy could be further enhanced by doing a filtering in the pool of possible classes as seen in Section 4.2. Razavian et al. [22] achieve the highest performance in the Oxford benchmark by applying a extensive spatial search at different scales for all images in the database. However, the cost of their feature extraction is much higher than ours since they feed 32 image crops of resolution 576×576 to the CNN. Our OnA proposal provides the third best result in the case of Oxford 5k, but the best in terms of descriptor dimension.

4.4 Re-Ranking and Query Expansion

A common approach in image retrieval systems is to apply some post-processing steps for refining a first fast search. We have explored query expansion and re-ranking, as they are common choices in the related work [15, 16, 31].

Query Expansion: There exist different ways to expand a visual query as introduced in [6, 7]. We choose one of the simplest and fastest ones as in [15], by simple updating the query descriptor for the l_2 normalized sum of the top ranked QE descriptors.

Local-aware Re-Ranking: As proposed in in [18], a first fast ranking based on the image features can be improved with a local analysis over the top retrieved R images. This re-ranking is based on a more detailed matching between the query object and the location of this object in each top-R ranked images. There are multiple ways to obtain object locations. For instance, R-MAC [31] applies a fast spatial search with approximate max-pooling localization. BoW [16] applies re-ranking using a sliding window approach with variable bounding boxes. Our approach, in contrast, localizes objects on the images using class activation maps as explained in Section 3.1. We use the most probable classes predicted from the query to generate the regions of interest in the target images. To obtain these regions, first we define heuristically a set of thresholds based on the normalized intensity of the CAM

Method	Dim	R	QE	Oxf5k	Par6k	Oxf105k	Par106k
CroW [15]	512	-	10	0.722	0.855	0.678	0.797
Ours(OnA)	512	-	10	0.760	0.873	-	-
Ours(OfA)	512	-	10	0.730	0.836	0.712	0.791
BoW [16]	25k	100	10	0.788	0.848	0.651	0.641
Ours(OnA)	512	100	10	0.780	0.874	-	-
Ours(OfA)	512	100	10	0.773	0.838	0.750	0.780
RMAC [31]	512	1000	5	0.770	**0.877**	0.726	**0.817**
Ours(OnA)	512	1000	5	**0.811**	0.874	-	-
Ours(OfA)	512	1000	5	0.801	0.855	**0.769**	0.800

(a)

Aggregation	Time (s)	mAP
Raw + PCA	0.5	0.420
1 CAM	0.5	0.667
8 CAMs	0.6	0.709
32 CAMs	0.9	0.711
64 CAMs	1.5	0.712

(b)

Table 2: (a) Comparison with the state-of-the-art after applying Re-Ranking (R) or/and Query Expansion (QE). Descriptor dimensions are included in the second column (Dim). (b) Computational burden after including CAMs.

heatmap values. More precisely, we define a set of values 1%, 10%, 20%, 30% and 40% of the max value of the CAM and compute bounding boxes around its largest connected component. Then, we build an image descriptor for every of the spatial regions and compare them with the query image using the cosine distance. We kept the one with the best score. The decision of using more than one threshold is to cover the variability of the objects dimensions in different images and detect with more precision. Using the average heatmap of the 2-top classes improved a bit the region generated, as most of the buildings are composed by more than one class. In Figure 2 we show an example of the regions of interest generated by this method.

We provide a comparison of our re-ranking and query expansion results with the rest of the relevant literature. CroW [15] authors only apply query expansion after the initial search while BoW and R-MAC apply first a spatial re-ranking. The number of top-images used to employ these techniques varies between works. For the sake of comparison, Table 2 provide our results with the same parameters for query expansion (QE) and re-ranking (R). For the initial search we keep N_{pca} of 1 and N_C of 64 for both OnA and OfA as in the previous section, but for the re-ranking we decrease N_C to the 6 more probable classes. This is done because after the first search we already have a set of relevant images and what we want is to perform a more fine-grained comparison by looking at particular regions. In addition to that, taking less classes reduces the computational time spent. Looking at Table 2, we can observe that our proposal achieves very competitive results even only with the application of a simple query expansion. If we add a re-ranking stage, we improve the performance mostly in Oxford dataset, where we obtain the top performance. In Paris, we can observe that performing the re-ranking step does not increase the performance mainly due to relevant images being already on the top QE of the ranked list.

5 Conclusions

In this work we proposed a technique to build compact representations for images focusing on their semantic content. To this end, we employed an image encoding pipeline that makes use of a pre-trained CNN and Class Activation Maps to extract discriminative regions from the image and weight its convolutional features accordingly. Our experiments showed the that selecting the relevant content of the image to build the descriptor is beneficial, and contributes to increase the retrieval performance. As a consequence, our approach outperforms other methods that build image representations combining off-the-shelf features focusing on random or fixed grid regions, establishing a new state-of-the-art.

References

[1] Jose Alvarez and Lars Petersson. Decomposeme: Simplifying convnets for end-to-end learning. *arXiv preprint arXiv:1606.05426*, 2016.

[2] Yannis Avrithis and Yannis Kalantidis. Approximate gaussian mixtures for large scale vocabularies. In *ECCV*, 2012.

[3] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, 2015.

[4] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014.

[5] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[6] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*. IEEE, 2007.

[7] Ondřej Chum, Andrej Mikulik, Michal Perdoch, and Jiří Matas. Total recall ii: Query expansion revisited. In *CVPR*, 2011.

[8] X Yu Felix, Rongrong Ji, Ming-Hen Tsai, Guangnan Ye, and Shih-Fu Chang. Weak attributes for large-scale image retrieval. In *CVPR*, 2012.

[9] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014.

[10] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *ECCV*, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[13] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Teddy Furon, and Ondrej Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. *arXiv preprint arXiv:1611.05113*, 2016.

[14] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *ECCV*. 2012.

[15] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.

[16] Eva Mohedano, Kevin McGuinness, Noel E O'Connor, Amaia Salvador, Ferran Marqués, and Xavier Giró-i Nieto. Bags of local convolutional features for scalable instance search. In *ICMR*, 2016.

[17] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.

[18] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[19] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[20] Filip Radenović, Hervé Jégou, and Ondrej Chum. Multiple measurements and joint dimensionality reduction for large scale image search with short vectors. In *ICMR*, 2015.

[21] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. *arXiv preprint arXiv:1604.02426*, 2016.

[22] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. A baseline for visual instance retrieval with deep convolutional networks. *arXiv preprint arXiv:1412.6574*, 2014.

[23] Cristian Reyes, Eva Mohedano, Kevin McGuinness, Noel E O'Connor, and Xavier Giro-i Nieto. Where is my phone?: Personal object retrieval from egocentric images. In *Proceedings of the first Workshop on Lifelogging Tools and Applications*. ACM, 2016.

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[25] Amaia Salvador, Xavier Giró-i Nieto, Ferran Marqués, and Shin'ichi Satoh. Faster r-cnn features for instance search. In *CVPR Workshop*, 2016.

[26] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, 2014.

[27] Xiaohui Shen, Zhe Lin, Jonathan Brandt, and Ying Wu. Spatially-constrained similarity measurefor large-scale object retrieval. *TPAMI*, 2014.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, 2016. URL http://arxiv.org/abs/1605.02688.

[30] Giorgos Tolias and Hervé Jégou. Visual query expansion with or without geometry: refining local descriptors by feature aggregation. *CVPR*, 2014.

[31] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *ICLR*, 2016.

[32] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

[33] Shiliang Zhang, Ming Yang, Xiaoyu Wang, Yuanqing Lin, and Qi Tian. Semantic-aware co-indexing for image retrieval. In *ICCV*, 2013.

[34] Bolei Zhou, Aditya Khosla, Lapedriza. Agata, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.